# Public Site Refactor Workflow - Session E

## Date Created: December 22, 2024

## 📊 Data Audit Summary

### Seeded Data (In Database) ☑

| Table | Count | Status |
|---|---|---|
| Services | 9 | ☑ Seeded via `seed-services.sql` |
| Rental Items | 55+ | ☑ Seeded via `seed-all-data.sql` |
| Rental Categories | 14 | ☑ Seeded in schema |
| Portfolio Projects | 5 | ☑ Seeded via `seed-all-data.sql` |
| FAQs | 12 | ☑ Seeded via `seed-all-data.sql` |
| FAQ Categories | 5 | ☑ Seeded in schema |
| Testimonials | 4 | ☑ Seeded via `seed-all-data.sql` |
| Venues | 4 | ☑ Seeded via `seed-all-data.sql` |
| Event Types | 6 | ☑ Seeded in schema |

### Missing Data (NOT Seeded) ❌

| Table | Status | Action Required |
|---|---|---|
| Team Members | ❌ NOT SEEDED | Create seed script or use CMS |
| Blog Posts | ❌ NOT SEEDED | Can be empty for V2.0, CMS ready |
| Blog Categories | ❌ NOT SEEDED | Create via CMS |
| Tags | ❌ NOT SEEDED | Create via CMS |

### Image URL Status

- Image URLs were updated via `update-image-urls.sql` to point to Supabase Storage
- Run verification query below to confirm

## 🔍 Verification Query

Run this query in Supabase SQL Editor to verify data:

```sql
-- Verify all seeded data counts
SELECT 'services' as table_name, COUNT(*) as count FROM services WHERE deleted_at
IS NULL
UNION ALL SELECT 'rental_items', COUNT(*) FROM rental_items WHERE deleted_at IS
NULL
UNION ALL SELECT 'rental_categories', COUNT(*) FROM rental_categories
UNION ALL SELECT 'portfolio_projects', COUNT(*) FROM portfolio_projects WHERE
deleted_at IS NULL
UNION ALL SELECT 'faqs', COUNT(*) FROM faqs WHERE deleted_at IS NULL
UNION ALL SELECT 'faq_categories', COUNT(*) FROM faq_categories
UNION ALL SELECT 'testimonials', COUNT(*) FROM testimonials WHERE deleted_at IS
NULL
UNION ALL SELECT 'venues', COUNT(*) FROM venues WHERE deleted_at IS NULL
UNION ALL SELECT 'event_types', COUNT(*) FROM event_types
UNION ALL SELECT 'team_members', COUNT(*) FROM team_members WHERE deleted_at IS
NULL
UNION ALL SELECT 'blog_posts', COUNT(*) FROM blog_posts WHERE deleted_at IS NULL
UNION ALL SELECT 'blog_categories', COUNT(*) FROM blog_categories
UNION ALL SELECT 'tags', COUNT(*) FROM tags;

-- Check for placeholder images (should return 0 for production-ready)
SELECT 'services_with_placeholder' as check_type, COUNT(*) FROM services
WHERE (thumbnail LIKE '%placeholder%' OR thumbnail IS NULL) AND deleted_at IS NULL
UNION ALL
SELECT 'rentals_with_placeholder', COUNT(*) FROM rental_items
WHERE thumbnail_url LIKE '%placeholder%' AND deleted_at IS NULL
UNION ALL
SELECT 'portfolio_with_placeholder', COUNT(*) FROM portfolio_projects
WHERE thumbnail_url LIKE '%placeholder%' AND deleted_at IS NULL;
```

---

## Page-by-Page Refactor Order

Following navbar display order:

### Phase 1: Home Page

- **Route:** `/` (`app/page.tsx`)
- **Data Needed:**
  - Core Services (hardcoded 4 items → keep static for featured curation)
  - Stats (hardcoded → keep static)
  - Testimonials (dynamic fetch, featured only)
- **Subcomponents to Extract:**
  - `components/home/stats-bar.tsx`
  - `components/home/core-services-section.tsx`
  - `components/home/who-we-serve-section.tsx`
  - `components/home/why-choose-us-section.tsx`
  - `components/home/cta-section.tsx`
  - `components/home/testimonials-section.tsx` (dynamic)

### Phase 2: Services

- **Routes:**
  - `/services` (`app/services/page.tsx`) - List all services
  - `/services/[slug]` (`app/services/[slug]/page.tsx`) - Dynamic service detail
- **Data Needed:**
  - All services with: use_cases (JSONB), process_steps (JSONB), packages (JSONB)
- **Note:** Current hardcoded service routes (e.g., `audio-visual-production/`) should redirect to dynamic `[slug]` route

### Phase 3: Rentals

- **Routes:**
  - `/rentals` (`app/rentals/page.tsx`) - Grid with category filter
  - `/rentals/[slug]` (`app/rentals/[slug]/page.tsx`) - Item detail
- **Data Needed:**
  - Rental items with category join
  - Rental categories for filter dropdown

### Phase 4: Portfolio

- **Routes:**
  - `/portfolio` (`app/portfolio/page.tsx`) - Grid with event type filter
  - `/portfolio/[slug]` (`app/portfolio/[slug]/page.tsx`) - Project detail
- **Data Needed:**
  - Portfolio projects with event_types join

### Phase 5: Venues

- **Route:** `/venues` (`app/venues/page.tsx`)
- **Data Needed:**
  - All active venues

### Phase 6: FAQs

- **Route:** `/faqs` (`app/faqs/page.tsx`)
- **Data Needed:**
  - FAQs grouped by category with category join

### Phase 7: About

- **Route:** `/about` (`app/about/page.tsx`)
- **Data Needed:**
  - Testimonials (featured, limit 3)
  - Team Members (future: when seeded)
- **Note:** Most content is static (values, timeline, mission)

### Phase 8: Blog (Lower Priority)

- **Routes:**

    - `/blog` - List posts
    - `/blog/[slug]` - Post detail
- **Data Needed:**
    - Blog posts with category and tags
- **Status:** CMS ready, pages can show "coming soon" or empty state

### Phase 9: Contact

- **Route:** `/contact` (`app/contact/page.tsx`)
- **Data Needed:**
    - Site settings (company info) - from DB or keep hardcoded for V2.0
    - Event types (for form dropdown)
- **Note:** Already has `/api/public/contact` endpoint

---

## 🏗 Architecture Pattern

### Layer Structure

```
Page (Server Component)
  └── Suspense Boundary + Skeleton Loading
        └── Data Fetching Component (Server)
              └── Public API Route → Supabase DB
```

### Example Implementation

```
// app/about/page.tsx
import { Suspense } from 'react';
import { TestimonialsSkeleton } from '@/components/skeletons';

export default function AboutPage() {
  return (
    <div>
      {/* Static content... */}

      <Suspense fallback={<TestimonialsSkeleton />}>
        <TestimonialsSection />
      </Suspense>
    </div>
  );
}

// components/about/testimonials-section.tsx (Server Component)
async function TestimonialsSection() {
  const res = await
fetch(`${process.env.NEXT_PUBLIC_SITE_URL}/api/public/testimonials?
featured=true&limit=3`, {
    next: { revalidate: 3600 } // Cache for 1 hour
  });
```

```
  const { data } = await res.json();

  return (
    <section>
      {data.map(testimonial => (
        <TestimonialCard key={testimonial.id} testimonial={testimonial} />
      ))}
    </section>
  );
}
```

---

## New Files to Create

### Public API Routes

```
app/api/public/services/route.ts              # GET all active services
app/api/public/services/[slug]/route.ts       # GET service by slug
app/api/public/rentals/route.ts               # GET rentals with category filter
app/api/public/rentals/[slug]/route.ts        # GET rental by slug
app/api/public/rental-categories/route.ts     # GET all categories
app/api/public/portfolio/route.ts             # GET portfolio projects
app/api/public/portfolio/[slug]/route.ts      # GET project by slug
app/api/public/venues/route.ts                # GET active venues
app/api/public/faqs/route.ts                  # GET FAQs by category
app/api/public/testimonials/route.ts          # GET featured testimonials
app/api/public/team/route.ts                  # GET active team members
app/api/public/event-types/route.ts           # GET event types (for forms)
```

### Skeleton Components

```
components/skeletons/services-grid-skeleton.tsx
components/skeletons/rentals-grid-skeleton.tsx
components/skeletons/portfolio-grid-skeleton.tsx
components/skeletons/testimonials-skeleton.tsx
components/skeletons/faqs-skeleton.tsx
components/skeletons/venues-skeleton.tsx
components/skeletons/service-detail-skeleton.tsx
components/skeletons/rental-detail-skeleton.tsx
components/skeletons/portfolio-detail-skeleton.tsx
components/skeletons/index.ts                 # Barrel export
```

### Home Page Components (Modularized)

```
components/home/stats-bar.tsx
components/home/core-services-section.tsx
```

```
components/home/who-we-serve-section.tsx
components/home/why-choose-us-section.tsx
components/home/cta-section.tsx
components/home/testimonials-section.tsx
components/home/index.ts
```

---

## ☑ Pre-Refactor Checklist

Before starting each phase:

1. ☐ Verify data exists in DB (run verification query)
2. ☐ Confirm image URLs are Supabase Storage URLs
3. ☐ Create public API route for the data
4. ☐ Create skeleton loading component
5. ☐ Refactor page to Server Component with Suspense
6. ☐ Test loading state and data display
7. ☐ Update types to use `Tables<'table_name'>`

---

## ⊘ Deferred Items

- ☐ Quote basket functionality (after full refactor)
- ☐ Contact form inquiry submission (after full refactor)
- ☐ Blog posts (can remain empty/coming soon)
- ☐ Team members section (needs seed data)

---

## 📝 Notes

### Type Pattern

```
import type { Tables } from '@/lib/supabase/types';

type Service = Tables<'services'>;
type Rental = Tables<'rental_items'> & {
  rental_categories: Tables<'rental_categories'> | null;
};
```

### Public API Pattern

```
// No authentication required for public routes
// Use direct Supabase server client
// Filter by is_active/deleted_at
// Cache responses with next: { revalidate: N }
```

### SEO Considerations

- Server-side rendering for all content pages
- Proper meta tags in layout/page
- Static generation where possible (generateStaticParams)
- Structured data (JSON-LD) for services and portfolio

---

## Session Breakdown

| Session | Focus | Priority |
|---|---|---|
| E1 | Home Page Modularization + Testimonials API | High |
| E2 | Services (List + Detail) | High |
| E3 | Rentals (List + Detail + Categories) | High |
| E4 | Portfolio (List + Detail) | High |
| E5 | Venues, FAQs, About | Medium |
| E6 | Blog, Contact Form Integration | Low |
| E7 | Cleanup + Remove Static Data | Final |