Training Program Investigation & Redesign

Date: November 3, 2025

Executive Summary

Investigation completed on the Mindful Champion training program system. The database contains proper data with 88 videos distributed across 3 programs, but there are implementation issues in the frontend components causing confusion and lack of value clarity.

Current State Analysis

Database Structure ✅ GOOD

- **3 Training Programs**:
  1. "2-Week Beginner Bootcamp" (14 days, Beginner) - 30 program videos
  2. "Dink Mastery Program" (10 days, Intermediate) - 29 program videos
  3. "Third Shot Excellence" (12 days, Advanced) - 29 program videos
- **88 Total Training Videos** - Professional YouTube-based training content
- **Proper Day Assignments** - Each day has 2-3 videos assigned
- **Database Schema** - Well-designed with TrainingProgram, TrainingVideo, ProgramVideo, UserProgram tables

Example Data Distribution (Third Shot Excellence):

- Day 1: 3 videos
- Day 2: 3 videos
- Day 3: 3 videos
- Day 4: 3 videos
- Day 5: 3 videos
- Day 6: 3 videos
- Day 7: 3 videos
- Day 8: 3 videos
- Day 9: 3 videos
- Day 10: 2 videos

Problems Identified

Problem 1: BootcampViewer Uses Hard-Coded Content ❌

Location: `/components/training/bootcamp-viewer.tsx`

Issue: The BootcampViewer component uses hard-coded content from `@/lib/bootcamp-content.ts` instead of the database videos. This means:
- The 30 videos assigned to the bootcamp in the database are IGNORED
- All content is manually written in code
- Changes require code deployments, not database updates
- Inconsistent with other programs

Code Evidence:

```
// bootcamp-viewer.tsx line 30
import { beginnerBootcampDays, type BootcampDay } from '@/lib/bootcamp-content'

// Line 40
const currentDayData = beginnerBootcampDays[selectedDay - 1]
// ^ Uses hard-coded array, ignores program.videos from database
```

Problem 2: Confusing User Journey

Issue: The training program flow has too many steps without clear value:
1. Homepage → “Start Training”
2. Training Hub → “Training Programs”
3. Program List → Select Program
4. Program Overview → “Start Program Now”
5. Daily Schedule → Click Day 1
6. Finally see actual content

User Feedback: “The training flow is confusing and wonky”

Problem 3: Lack of Value Clarity

Issue: When users view the program overview or daily schedule, they don’t immediately see:
- What specific drills they’ll do
- What videos they’ll watch
- The concrete value they’re getting

Evidence: User screenshot shows “0 drills” perception - even though videos exist in database

---

Solution Design

Phase 1: Unified Program Viewer ✅

Create ONE unified program viewer that:
- Uses DATABASE videos (not hard-coded content)
- Works for ALL programs (Beginner, Intermediate, Advanced)
- Shows clear daily structure with video count
- Provides engaging, valuable presentation

Phase 2: Enhanced Daily Structure

For each day, clearly show:

```
Day X: [Focus Area]
├─ 2-3 Training Videos (with thumbnails & titles)
├─ Practice Guidelines (generated from video topics)
├─ Estimated Time
└─ Learning Objectives
```

Phase 3: Simplified User Journey

Streamline the flow:
1. Training Hub → Shows programs with preview
2. Select Program → Immediately see daily curriculum
3. Click any day → Start training with videos

Phase 4: Value Proposition Enhancement

Make it crystal clear what users get:
- “88 Professional Training Videos”
- “Structured X-Day Program”
- “3-5 Videos Per Day”
- “Track Your Progress”
- Clear outcomes and benefits

Implementation Plan

Step 1: Create Universal Program Viewer

- Merge best features of BootcampViewer and EnterpriseProgramViewer
- Use database videos for ALL programs
- Add rich daily structure with clear value
- Include progress tracking
- Mobile-responsive design

Step 2: Enhance Daily Content Display

```typescript
interface DailyContent {
  day: number
  focus: string  // e.g., "Serve Fundamentals"
  videos: Video[]  // From database
  estimatedTime: string  // Calculated from video durations
  objectives: string[]  // Generated from video topics/keywords
  practiceGuidelines: string[]  // Smart defaults
}
```

Step 3: Update Program Data Structure

Ensure `dailyStructure` JSON in database aligns with video assignments:

```
{
  "day1": {
    "focus": "Serve Fundamentals",
    "objectives": [...],
    "practice": "..."
  }
}
```

Step 4: Improve Navigation

- Add breadcrumbs
- Clear “Back” buttons
- Direct deep linking to days
- Save scroll position

Database Status: ALREADY GOOD ✅

The database does NOT need changes. Data is properly structured:

```
TrainingProgram (3 programs)
  ├── ProgramVideo (links to videos by day)
  │     ├── Day 1: Videos 1,2,3
  │     ├── Day 2: Videos 4,5,6
  │     └── ...
  └── TrainingVideo (88 videos)
```

What we need to do is make the **frontend components USE this data properly**.

Benefits of This Approach

1. **Consistency**: All programs use the same data source (database)
2. **Flexibility**: Easy to update program content without code changes
3. **Scalability**: Add new programs easily
4. **Clarity**: Users see exactly what they’re getting
5. **Value**: Crystal clear value proposition
6. **Maintainability**: One source of truth (database)

Next Steps

1. ✅ Complete investigation
2. 🔄 Create universal program viewer component
3. ⏳ Update program pages to use new viewer
4. ⏳ Add daily content guidelines
5. ⏳ Test complete user journey
6. ⏳ Deploy and monitor user feedback

Technical Details

Current Program Fetch (CORRECT):

```
// app/train/programs/[programId]/page.tsx
const program = await prisma.trainingProgram.findUnique({
  where: { programId: params.programId },
  include: {
    programVideos: {
      include: { video: true },
      orderBy: [{ day: 'asc' }, { order: 'asc' }]
    }
  }
})
```

This correctly fetches all videos with their day assignments. The problem is the viewer components don’t use this data consistently.

Conclusion

The training program system has a solid foundation with proper database structure and content. The fix is primarily frontend - creating a unified, data-driven viewer that clearly presents value and makes the user journey intuitive.

Status: Ready to implement Phase 1 - Universal Program Viewer