PRD — TeamOpsHQ (Youth Team Management) -
Mobile-First Implementation Update
Version: 2025-08-17 • Owner: You • Status: Cross-platform strategy revision

1. Problem & Goals (Mobile Context Added)

Problem (Mobile-Refined)

Youth sports organizations face a three-way complexity problem that is **fundamentally mobile**:

1. **Coaches need quick, reliable field tools** that work offline on phones/tablets and don't interrupt practice flow
2. **Parents need consolidated, child-specific information** accessible on mobile during busy family schedules
3. **Admins need operational control** with compliance safeguards that work from anywhere, including sidelines

Current solutions force platform trade-offs: web-only tools don't work on fields, native apps are expensive to maintain, and nothing properly handles offline-first mobile workflows for youth organizations.

Mobile Success Metrics Added:

- **Coach Mobile Adoption**: 95% of note capture happens on mobile devices
- **Offline Reliability**: 99% of offline actions sync successfully on reconnect
- **Mobile Performance**: <1s app startup, <200ms note capture response
- **Cross-Platform Consistency**: Feature parity between web and mobile within 2 versions

2. Platform Strategy (New Section)

Cross-Platform Architecture Decision

Recommended Approach: Progressive Web App (PWA) + React Native

Phase 1: PWA-First (Weeks 1-8)

- Single codebase serves web + mobile web
- Install-to-homescreen for app-like experience
- Full offline capability via Service Worker
- Push notifications via web APIs

- Camera access for photo attachments

Phase 2: React Native App (Weeks 12-20)

- Shared business logic with web via shared packages
- Native performance for complex interactions
- App Store distribution for broader adoption
- Better offline data management
- Native push notifications and background sync

Why PWA First?

1. **Solo Developer Efficiency**: One codebase, deploy everywhere
2. **Faster Pilot Validation**: No app store approval delays
3. **Lower Barrier to Entry**: No download friction for parents
4. **Easier Updates**: Instant deployment vs app store reviews
5. **Cross-Platform by Default**: Works on iOS, Android, desktop

PWA Capabilities Assessment:

✅ **Offline-first note capture** - Service Worker + IndexedDB ✅ **Push notifications** - Web Push API (Android + Desktop, iOS 16.4+) ✅ **Camera access** - Web APIs for photo capture ✅ **Install to homescreen** - Native app feel ✅ **Background sync** - Service Worker background sync ⚠️ **iOS limitations** - Some features require iOS 16.4+, no background app refresh

3. Technology Stack (Updated)

Core Cross-Platform Stack

Frontend Framework:

- **Next.js 15** with App Router + PWA plugin
- **React Query** for server state management
- **Zustand + Persist** for offline state
- **Tailwind CSS** for responsive mobile-first design

Backend:

- **Next.js API Routes** (or separate Express.js if needed for React Native)
- **PostgreSQL** + **Prisma ORM**

- **Clerk** for authentication (supports both web and React Native)

Mobile-Specific Additions:

typescript

```typescript
// PWA Configuration
const withPWA = require('next-pwa')({
  dest: 'public',
  register: true,
  skipWaiting: true,
  runtimeCaching: [
    {
      urlPattern: /^https:\/\/api\.teamopshq\.com\/.*$/,
      handler: 'StaleWhileRevalidate',
      options: {
        cacheName: 'api-cache',
        expiration: {
          maxEntries: 100,
          maxAgeSeconds: 60 * 60 * 24 // 24 hours
        }
      }
    }
  ]
})

// Offline-first state management
const useOfflineNotes = create(
  persist(
    (set, get) => ({
      notes: [],
      pendingSync: [],
      addNote: (note) => {
        // Optimistic update + queue for sync
        set(state => ({
          notes: [...state.notes, note],
          pendingSync: [...state.pendingSync, note]
        }))
      }
    }),
    { name: 'offline-notes-storage' }
  )
)
```

Offline Strategy (Mobile-Optimized):

- **Service Worker** for app shell caching
- **IndexedDB** via Dexie.js for structured offline data
- **Background Sync API** for automatic retry
- **Optimistic updates** with rollback capability
- **Network-first/Cache-first** strategies per data type

Mobile UI Framework:

```typescript
// Mobile-optimized component library
import {
  Sheet,
  Drawer,
  SwipeableList,
  PullToRefresh,
  BottomTabs
} from '@/components/mobile'

// Touch-optimized interactions
const CoachNotesScreen = () => {
  return (
    <PullToRefresh onRefresh={syncNotes}>
      <SwipeableList
        items={athletes}
        onSwipe={quickAddNote}
        renderItem={AthleteCard}
      />
      <BottomSheet>
        <QuickTagSelector />
      </BottomSheet>
    </PullToRefresh>
  )
}
```

4. User Experience (Mobile-First Design)

Coach Mobile Workflow:

1. **Quick Launch**: PWA icon on home screen, <1s startup

2. **Roster Overview**: Large touch targets, athlete photos
3. **One-Tap Note Entry**: Swipe athlete card → quick tags appear
4. **Bulk Actions**: Multi-select with checkboxes for group operations
5. **Offline Indicator**: Clear sync status, retry failed uploads
6. **Voice Notes**: Optional voice-to-text for hands-free capture

Parent Mobile Experience:

1. **Child-Focused Dashboard**: Swipe between multiple children
2. **Quick RSVP**: One-tap responses with optional notes
3. **Photo Sharing**: Camera integration for team photos
4. **Push Notifications**: Practice reminders, coach updates
5. **Offline Reading**: Cached notes and announcements

Mobile-Specific Features:

- **Haptic Feedback**: Confirmation for important actions
- **Dark Mode**: Automatic switching based on system preference
- **Large Touch Targets**: Minimum 44px tap areas
- **Swipe Gestures**: Natural mobile navigation patterns
- **Pull-to-Refresh**: Standard mobile data refresh pattern

5. Implementation Phases (Mobile-Updated)

Phase 1A: PWA Foundation (Weeks 1-2)

Mobile-First Setup:

- Next.js + PWA configuration
- Responsive breakpoints: mobile (390px), tablet (768px), desktop (1024px)
- Touch-optimized UI components
- Service Worker for offline capability
- Basic push notification setup

Deliverables:

- Installable PWA with offline shell
- Mobile-responsive authentication flow

- Touch-optimized navigation

Phase 1B: Coach Mobile Tools (Weeks 3-4)

Offline-First Note Capture:

- Athlete roster with photo thumbnails and large touch targets
- Swipe gestures for quick note entry
- Offline storage with visual sync indicators
- Camera integration for progress photos
- Voice-to-text note entry

Mobile-Specific Enhancements:

typescript

```typescript
// Touch-optimized athlete selection
const AthleteGrid = () => {
  return (
    <div className="grid grid-cols-2 md:grid-cols-3 gap-4 p-4">
      {athletes.map(athlete => (
        <TouchableCard
          key={athlete.id}
          onTap={() => selectAthlete(athlete)}
          onLongPress={() => showQuickActions(athlete)}
          className="aspect-square"
        >
          <AthletePhoto src={athlete.photo} />
          <span className="text-lg font-semibold">{athlete.jerseyNumber}</span>
        </TouchableCard>
      ))}
    </div>
  )
}
```

Phase 1C: Parent Mobile Experience (Weeks 5-6)

Mobile-Optimized Parent Flow:

- Swipe-based onboarding wizard
- Child selection with large profile cards
- Pull-to-refresh for updates

- Native share functionality for team info
- Camera permissions for profile photos

Phase 1D: PWA Polish & Admin Mobile (Weeks 7-8)

Cross-Device Admin Experience:

- Responsive admin dashboard
- Mobile-friendly data tables
- Touch-optimized form inputs
- Offline-capable compliance checklist

6. Data Architecture (Mobile-Optimized)

Offline-First Data Strategy:

typescript

```typescript
// Mobile data sync patterns
enum SyncStrategy {
  IMMEDIATE = 'immediate',    // Notes, attendance
  BATCHED = 'batched',       // Analytics, logs
  BACKGROUND = 'background',  // Photos, media
  ON_DEMAND = 'on_demand'    // Historical data
}

// Mobile storage quotas
const STORAGE_LIMITS = {
  PHOTOS: 50 * 1024 * 1024,     // 50MB for athlete photos
  NOTES: 10 * 1024 * 1024,      // 10MB for text notes
  CACHE: 100 * 1024 * 1024,     // 100MB for app cache
}
```

Mobile-Specific Schema Additions:

sql

```
-- Device and sync tracking
CREATE TABLE device_sessions (
  id UUID PRIMARY KEY,
  user_id UUID REFERENCES users(id),
  device_info JSONB, -- Device type, OS, app version
  last_sync TIMESTAMP,
  offline_actions_pending INTEGER DEFAULT 0
);

-- Photo and media handling
CREATE TABLE media_attachments (
  id UUID PRIMARY KEY,
  entity_type TEXT, -- 'note', 'athlete_profile', 'team_photo'
  entity_id UUID,
  file_path TEXT,
  thumbnail_path TEXT,
  uploaded_from_mobile BOOLEAN DEFAULT false,
  sync_status TEXT DEFAULT 'pending'
);
```

7. Performance & Offline Strategy (Mobile-Specific)

Mobile Performance Targets:

- **App Startup**: <1s cold start, <300ms warm start
- **Note Capture**: <100ms tap-to-local-save
- **Photo Upload**: Background with progress indicator
- **Sync on Reconnect**: <5s for typical practice worth of notes
- **Battery Usage**: <5% battery drain per 2-hour practice

Offline Capabilities by Feature:

typescript

```
// Offline capability matrix
const OFFLINE_FEATURES = {
  noteCapture: 'FULL',       // Complete offline functionality
  attendance: 'FULL',        // Mark attendance offline
  rsvp: 'FULL',              // Parent responses cached
  chat: 'READ_ONLY',         // View cached messages
  photoUpload: 'QUEUED',     // Upload when online
  adminReports: 'NONE'       // Requires real-time data
}
```

Mobile-Specific Sync Logic:

typescript

```
const useMobileSync = () => {
  const { isOnline } = useNetworkState()
  const { data: pendingActions } = usePendingSyncActions()

  useEffect(() => {
    if (isOnline && pendingActions.length > 0) {
      syncPendingActions({
        priority: 'notes_first', // Notes before photos
        batchSize: 10,
        retryAttempts: 3
      })
    }
  }, [isOnline])
}
```

8. Future React Native Migration (Phase 2)

Shared Code Strategy:

typescript

```
// Packages structure for code sharing
packages/
├── shared-types/        # TypeScript definitions
├── api-client/          # API calls and data fetching
├── business-logic/      # Pure functions, validation
├── ui-components/       # Platform-agnostic components
└── utils/               # Date formatting, calculations

apps/
├── web/                 # Next.js PWA
├── mobile/              # React Native app
└── api/                 # Backend (if separated)
```

Migration Benefits:

- **Better Performance**: Native rendering and animations
- **Enhanced Offline**: SQLite integration, better background processing
- **Platform Features**: Better camera, contacts integration, native notifications
- **App Store Presence**: Professional app distribution
- **Advanced Offline**: Background sync, local database replication

9. Risk Mitigation (Mobile-Added)

Mobile-Specific Risks:

PWA Limitations:

- iOS Safari restrictions on PWA features
- Battery optimization affecting background sync
- Storage quota limitations on mobile devices
- Network connectivity issues at field locations

Mitigation Strategies:

- Progressive enhancement: core features work even with limitations
- Aggressive caching of critical data (rosters, recent notes)
- Clear offline indicators and manual sync triggers
- Fallback to SMS for critical notifications if push fails

Cross-Platform Complexity:

- Testing on multiple devices and browsers
- Platform-specific UI/UX patterns
- Sync conflicts between web and mobile usage

10. Solo Developer Mobile Strategy

MVP Mobile Approach:

1. **Start PWA-only**: Single codebase, faster iteration
2. **Mobile-first design**: All features designed for touch first
3. **Progressive enhancement**: Add native app later based on adoption
4. **Community testing**: Use pilot team for real device testing

Development Tools for Mobile:

- **Responsive design**: Chrome DevTools device simulation
- **PWA testing**: Lighthouse PWA audits
- **Real device testing**: BrowserStack for cross-device testing
- **Offline testing**: Chrome Network throttling and offline simulation

Immediate Next Actions:

1. Set up Next.js with PWA plugin and mobile-first Tailwind config
2. Build touch-optimized coach notes interface
3. Test offline functionality on actual mobile devices
4. Deploy to Vercel with PWA manifest for installation testing
5. Get pilot coaches using PWA on their phones within week 2

Mobile Success Criteria for Pilot:

- All coaches install PWA to home screen within first practice
- 100% of notes captured on mobile devices (not desktop)
- Zero complaints about offline functionality during practices
- Parents actively use mobile RSVP features

This mobile-first approach ensures your core users (coaches on sidelines, parents on-the-go) get the experience they need while maintaining development efficiency for a solo developer.