🚀 PANA SPORTS V2.0 - IMPLEMENTATION WORKFLOW

📋 Prerequisites

Before starting implementation:

1. **Run the database migration** on Supabase Dashboard:
    - Go to SQL Editor
    - Open and run: `supabase/migrations/20241231_v2_0_migration.sql`
    - Verify tables created successfully
2. **Regenerate TypeScript types** (after migration):

```
# // turbo
npx supabase gen types typescript --project-id YOUR_PROJECT_ID >
lib/supabase/database.types.ts
```

PHASE 1: Core Infrastructure

Task 1.1: Create Season Types & Schemas

Create `lib/schemas/season.ts`:

```
import { z } from 'zod';

export const seasonCreateSchema = z.object({
  name: z.string().min(1),
  slug: z.string().min(1),
  start_date: z.string(),
  end_date: z.string(),
  is_current: z.boolean().optional().default(false),
  description_en: z.string().optional(),
  description_am: z.string().optional(),
});

export const seasonUpdateSchema = seasonCreateSchema.partial();

export type SeasonCreate = z.infer<typeof seasonCreateSchema>;
export type SeasonUpdate = z.infer<typeof seasonUpdateSchema>;
```

Task 1.2: Create Season Hooks

Create `lib/hooks/public/useSeasons.ts`:

```
import { useQuery } from '@tanstack/react-query';

export interface Season {
  id: string;
  name: string;
  slug: string;
  start_date: string;
  end_date: string;
  is_current: boolean;
  is_archived: boolean;
  description_en?: string;
  description_am?: string;
  created_at: string;
}

export function useSeasons() {
  return useQuery<Season[]>({
    queryKey: ['seasons'],
    queryFn: async () => {
      const res = await fetch('/api/public/seasons');
      if (!res.ok) throw new Error('Failed to fetch seasons');
      return res.json();
    },
  });
}

export function useCurrentSeason() {
  return useQuery<Season>({
    queryKey: ['seasons', 'current'],
    queryFn: async () => {
      const res = await fetch('/api/public/seasons/current');
      if (!res.ok) throw new Error('Failed to fetch current season');
      return res.json();
    },
  });
}

export function useSeason(id: string) {
  return useQuery<Season>({
    queryKey: ['seasons', id],
    queryFn: async () => {
      const res = await fetch(`/api/public/seasons/${id}`);
      if (!res.ok) throw new Error('Failed to fetch season');
      return res.json();
    },
    enabled: !!id,
  });
}
```

Task 1.3: Create Season API Routes

Create `app/api/public/seasons/route.ts`:

```
import { createClient } from '@/lib/supabase/server';
import { NextResponse } from 'next/server';

export async function GET() {
  try {
    const supabase = await createClient();
    const { data, error } = await supabase
      .from('seasons')
      .select('*')
      .order('start_date', { ascending: false });

    if (error) throw error;
    return NextResponse.json(data);
  } catch (error) {
    console.error('Error fetching seasons:', error);
    return NextResponse.json({ error: 'Failed to fetch seasons' }, { status: 500
});
  }
}
```

Create `app/api/public/seasons/current/route.ts`:

```
import { createClient } from '@/lib/supabase/server';
import { NextResponse } from 'next/server';

export async function GET() {
  try {
    const supabase = await createClient();
    const { data, error } = await supabase
      .from('seasons')
      .select('*')
      .eq('is_current', true)
      .single();

    if (error) throw error;
    return NextResponse.json(data);
  } catch (error) {
    console.error('Error fetching current season:', error);
    return NextResponse.json({ error: 'Failed to fetch current season' }, {
status: 500 });
  }
}
```

PHASE 2: Main Site UI Updates

Task 2.1: Navbar Enhancement

Modify `components/shared/navbar.tsx`:

Changes to make:

1. Increase logo height from `h-12` to `h-16 lg:h-20`
2. Move Women's League into League dropdown
3. Add "Features" link for opinion articles

Key sections to update:

```
// Line ~327-335: Increase logo size
<Image
  src="/logo1.png"
  alt="Pana Sports"
  width={160}  // Increased from 120
  height={64}  // Increased from 48
  className="h-16 lg:h-20 w-auto object-contain transition-transform duration-300 group-hover:scale-105"
  priority
/>

// Line ~96-122: Update navItems
const navItems: NavItem[] = [
  { href: "/", label: t.home, type: "link" },
  { href: "/news", label: t.news, type: "link" },
  { href: "/features", label: t.features, type: "link" }, // NEW
  {
    label: t.league,
    type: "dropdown",
    items: [
      { href: "/premier-league", label: t.pl },
      { href: "/ethiopian-cup", label: t.ec }, // Update to cups page later
      { href: "/higher-league", label: t.hl },
      { href: "/league-one", label: t.lo },
      { href: "/womens-league", label: t.womens }, // MOVED HERE
    ],
  },
  // { href: "/womens-league", label: t.womens, type: "link" }, // REMOVE THIS
  {
    label: t.national,
    type: "dropdown",
    // ... keep items as is
  },
  { href: "/athletics", label: t.athletics, type: "link" },
];

// Add to labels object:
features: "Features", // English
features: "ትንታኔዎች", // Amharic
```

Task 2.2: Dynamic AdBanner Component

Update `components/shared/AdBanner.tsx`:

```
"use client";

import { useState, useEffect } from "react";
import Image from "next/image";
import { ChevronLeft, ChevronRight, X } from "lucide-react";
import { motion, AnimatePresence } from "framer-motion";
import { Button } from "@/components/ui/button";
import { cn } from "@/lib/utils";

interface AdBannerProps {
  variant?: 'full' | 'sidebar' | 'inline';
  height?: string;
  showControls?: boolean;
  showClose?: boolean;
  className?: string;
  page?: string; // For tracking which page the ad is on
}

export default function AdBanner({
  variant = 'full',
  height,
  showControls = true,
  showClose = true,
  className,
  page = 'home'
}: AdBannerProps) {
  const [currentAd, setCurrentAd] = useState(0);
  const [isVisible, setIsVisible] = useState(true);

  // TODO: Replace with useAds hook when API is ready
  const ads = [
    { id: 1, image: "/ad1.jpg", alt: "Advertisement 1" },
    { id: 2, image: "/ad2.png", alt: "Advertisement 2" },
    { id: 3, image: "/ad3.jpg", alt: "Advertisement 3" },
  ];

  useEffect(() => {
    const interval = setInterval(() => {
      setCurrentAd((prev) => (prev + 1) % ads.length);
    }, 5000);
    return () => clearInterval(interval);
  }, [ads.length]);

  const nextAd = () => setCurrentAd((prev) => (prev + 1) % ads.length);
  const prevAd = () => setCurrentAd((prev) => (prev - 1 + ads.length) %
ads.length);

  if (!isVisible) return null;

  // Variant-based styling
  const variantStyles = {
    full: "w-full h-24 md:h-32",
    sidebar: "w-full h-48 md:h-64",
    inline: "w-full h-20 md:h-24",
  };

  const containerHeight = height || variantStyles[variant];

  return (
    <motion.div
      initial={{ opacity: 0, y: -20 }}
      animate={{ opacity: 1, y: 0 }}
      exit={{ opacity: 0, y: -20 }}
      className={cn(
        "relative bg-zinc-900/30 backdrop-blur-md border border-white/5 rounded-xl
overflow-hidden",
        className
      )}
    >
      <div className={cn("relative", containerHeight)}>
        <AnimatePresence mode="wait">
          <motion.div
            key={currentAd}
            initial={{ opacity: 0, x: 100 }}
            animate={{ opacity: 1, x: 0 }}
            exit={{ opacity: 0, x: -100 }}
            transition={{ duration: 0.5 }}
            className="absolute inset-0"
          >
            <Image
              src={ads[currentAd].image}
              alt={ads[currentAd].alt}
              fill
              className="object-cover"
              priority={currentAd === 0}
            />
          </motion.div>
        </AnimatePresence>

        {/* Controls */}
        {showControls && (
          <>
            <Button
              variant="ghost"
              size="icon"
              onClick={prevAd}
              className="absolute left-2 top-1/2 -translate-y-1/2 h-8 w-8 rounded-
full bg-black/50 hover:bg-black/70 border border-white/10"
            >
              <ChevronLeft className="h-4 w-4" />
            </Button>
            <Button
              variant="ghost"
              size="icon"
              onClick={nextAd}
              className="absolute right-2 top-1/2 -translate-y-1/2 h-8 w-8
rounded-full bg-black/50 hover:bg-black/70 border border-white/10"
            >
              <ChevronRight className="h-4 w-4" />
            </Button>
          </>
        )}

        {/* Indicators */}
        <div className="absolute bottom-2 left-1/2 -translate-x-1/2 flex items-
center gap-1.5">
          {ads.map((_, idx) => (
            <button
              key={idx}
              onClick={() => setCurrentAd(idx)}
              className={cn(
                "h-1.5 rounded-full transition-all duration-300",
                idx === currentAd ? "w-6 bg-primary" : "w-1.5 bg-white/30
hover:bg-white/50"
              )}
            />
          ))}
        </div>

        {/* Close */}
        {showClose && (
          <Button
            variant="ghost"
            size="icon"
            onClick={() => setIsVisible(false)}
            className="absolute top-2 right-2 h-6 w-6 rounded-full bg-black/50
hover:bg-black/70 text-zinc-400 hover:text-white"
          >
            <X className="h-3 w-3" />
          </Button>
        )}
      </div>
    </motion.div>
  );
}
```

Task 2.3: Update Home Page with Ad Placements

Modify `app/page.tsx`:

```
// app/page.tsx
import AdBanner from "@/components/shared/AdBanner";
import HomeNewsSection from "@/components/news/HomeNewsSection";
import RightColumn from "@/components/shared/RightColumn";
import HomeNewsSectionSkeleton from
"@/components/shared/Skeletons/HomeNewsSectionSkeleton";
```

```
import RightColumnSkeleton from
"@/components/shared/Skeletons/RightColumnSkeleton";
import OtherNews from "@/components/news/OtherNews"; // NEW
import { Suspense } from "react";

export default function Home() {
  return (
    <>
      {/* Top Ad Banner */}
      <AdBanner variant="full" page="home" />

      <main className="min-h-screen bg-black text-white pt-8 pb-10">
        <div className="container mx-auto px-4 space-y-6">
          {/* Main Content Grid */}
          <div className="grid grid-cols-1 lg:grid-cols-12 gap-6 items-start">
            {/* Left Column - News Section */}
            <div className="lg:col-span-8 space-y-6">
              <Suspense fallback={<HomeNewsSectionSkeleton />}>
                <HomeNewsSection />
              </Suspense>

              {/* Ad after news feed */}
              <AdBanner variant="inline" showClose={false} page="home" />

              {/* Other News Grid (2x3) */}
              <Suspense fallback={<div className="h-48 bg-zinc-900 animate-pulse
rounded-xl" />}>
                <OtherNews />
              </Suspense>
            </div>

            {/* Right Column */}
            <div className="lg:col-span-4 space-y-6">
              <Suspense fallback={<RightColumnSkeleton />}>
                <RightColumn />
              </Suspense>
            </div>
          </div>
        </div>
      </main>
    </>
  );
}
```

Task 2.4: Create OtherNews Component

Create `components/news/OtherNews.tsx`:

```
"use client";

import { useHomeNews } from "@/lib/hooks/public/useNews";
import { transformNewsList } from "@/lib/utils/transformers";
import Link from "next/link";
import Image from "next/image";
import { motion } from "framer-motion";
import { ArrowRight } from "lucide-react";

export default function OtherNews() {
  const { data: news, isLoading } = useHomeNews();

  if (isLoading) return null;

  // Get news items after the first 5 (which are shown in main section)
  const transformedNews = transformNewsList(news || []);
  const otherNews = transformedNews.slice(5, 11); // Items 6-11 for 2x3 grid

  if (otherNews.length === 0) return null;

  return (
    <section className="bg-zinc-900/40 backdrop-blur-xl border border-white/5
rounded-2xl p-6">
      <div className="flex items-center justify-between mb-6">
        <h3 className="text-lg font-bold text-white">More Stories</h3>
        <Link
          href="/news"
          className="text-sm text-primary hover:text-primary/80 flex items-center
gap-1"
        >
          View All <ArrowRight className="w-4 h-4" />
        </Link>
      </div>

      <div className="grid grid-cols-2 md:grid-cols-3 gap-4">
        {otherNews.map((item, idx) => (
          <motion.div
            key={item.id}
            initial={{ opacity: 0, y: 20 }}
            animate={{ opacity: 1, y: 0 }}
            transition={{ delay: idx * 0.1 }}
          >
            <Link
              href={`/news/${item.slug}`}
              className="group block"
            >
              <div className="relative aspect-video rounded-lg overflow-hidden mb-
2">
                <Image
                  src={item.thumbnail || '/placeholder.jpg'}
                  alt={item.title}
                  fill
                  className="object-cover transition-transform duration-300 group-
hover:scale-105"
                />
              </div>
              <h4 className="text-sm font-medium text-zinc-200 line-clamp-2 group-
hover:text-primary transition-colors">
                {item.title}
              </h4>
              <p className="text-xs text-zinc-500 mt-1">
                {item.category}
              </p>
            </Link>
          </motion.div>
        ))}
      </div>
    </section>
  );
}
```

Task 2.5: Remove Standings Limit

Modify `components/shared/RightColumn.tsx`:

```
// Find line ~59-62 and remove the limit
const {
  data: standings,
  isLoading: isLoadingStandings,
  isError: isErrorStandings,
} = useStandings({
  league_id: leagues?.[currentLeagueIndex]?.id,
  // limit: 6,  // REMOVE THIS LINE
});
```

Also update `components/standings/StandingsTable.tsx` to not show "View Full Standings" button when showing all:

```
// Add a prop to control this
interface StandingsTableProps {
  // ... existing props
  showViewAllButton?: boolean;
  maxDisplay?: number; // NEW: optional limit for display
}

// In the component, conditionally render based on whether all teams are shown
{showViewAllButton && maxDisplay && standings.length > maxDisplay && (
  <Button>View Full Standings</Button>
)}
```

Task 2.6: Global Search Component

Create `components/shared/GlobalSearch.tsx`:

```
"use client";

import { useState, useEffect, useRef } from "react";
import { Search, X, Users, User, Newspaper, Calendar } from "lucide-react";
import { Input } from "@/components/ui/input";
import { Button } from "@/components/ui/button";
import { motion, AnimatePresence } from "framer-motion";
import { useDebounce } from "@/lib/hooks/useDebounce";
import Link from "next/link";
import Image from "next/image";
import { cn } from "@/lib/utils";

interface SearchResult {
  type: 'team' | 'player' | 'news' | 'match';
  id: string;
  title: string;
  subtitle?: string;
  image?: string;
  href: string;
}

export default function GlobalSearch() {
  const [isOpen, setIsOpen] = useState(false);
  const [query, setQuery] = useState("");
  const [results, setResults] = useState<SearchResult[]>([]);
  const [isLoading, setIsLoading] = useState(false);
  const inputRef = useRef<HTMLInputElement>(null);
  const debouncedQuery = useDebounce(query, 300);

  // Keyboard shortcut
  useEffect(() => {
    const handleKeyDown = (e: KeyboardEvent) => {
      if ((e.metaKey || e.ctrlKey) && e.key === 'k') {
        e.preventDefault();
        setIsOpen(true);
      }
      if (e.key === 'Escape') {
        setIsOpen(false);
      }
    };
    window.addEventListener('keydown', handleKeyDown);
    return () => window.removeEventListener('keydown', handleKeyDown);
  }, []);

  // Focus input when opened
  useEffect(() => {
    if (isOpen && inputRef.current) {
      inputRef.current.focus();
    }
  }, [isOpen]);

  // Search API call
  useEffect(() => {
    if (debouncedQuery.length < 2) {
      setResults([]);
      return;
    }

    const search = async () => {
      setIsLoading(true);
      try {
        const res = await fetch(`/api/public/search?
q=${encodeURIComponent(debouncedQuery)}`);
        if (res.ok) {
          const data = await res.json();
          setResults(data);
        }
      } catch (error) {
        console.error('Search error:', error);
      } finally {
        setIsLoading(false);
      }
    };

    search();
  }, [debouncedQuery]);

  const getIcon = (type: string) => {
    switch (type) {
      case 'team': return <Users className="w-4 h-4 text-blue-400" />;
      case 'player': return <User className="w-4 h-4 text-green-400" />;
      case 'news': return <Newspaper className="w-4 h-4 text-purple-400" />;
      case 'match': return <Calendar className="w-4 h-4 text-orange-400" />;
      default: return null;
    }
  };

  return (
    <>
      {/* Search Trigger Button */}
      <Button
        variant="ghost"
        size="sm"
        onClick={() => setIsOpen(true)}
        className="hidden lg:flex items-center gap-2 text-zinc-400 hover:text-
white"
      >
        <Search className="w-4 h-4" />
        <span className="text-sm">Search</span>
        <kbd className="ml-2 text-xs bg-zinc-800 px-1.5 py-0.5 rounded">⌘K</kbd>
      </Button>

      {/* Search Overlay */}
      <AnimatePresence>
        {isOpen && (
          <>
            {/* Backdrop */}
```

```
              <motion.div
                initial={{ opacity: 0 }}
                animate={{ opacity: 1 }}
                exit={{ opacity: 0 }}
                onClick={() => setIsOpen(false)}
                className="fixed inset-0 bg-black/60 backdrop-blur-sm z-50"
              />

              {/* Search Modal */}
              <motion.div
                initial={{ opacity: 0, scale: 0.95 }}
                animate={{ opacity: 1, scale: 1 }}
                exit={{ opacity: 0, scale: 0.95 }}
                className="fixed top-[15%] left-1/2 -translate-x-1/2 w-full max-w-
2xl z-50 p-4"
              >
                <div className="bg-zinc-900 border border-zinc-800 rounded-2xl
shadow-2xl overflow-hidden">
                  {/* Search Input */}
                  <div className="flex items-center gap-3 p-4 border-b border-zinc-
800">
                    <Search className="w-5 h-5 text-zinc-400" />
                    <Input
                      ref={inputRef}
                      value={query}
                      onChange={(e) => setQuery(e.target.value)}
                      placeholder="Search teams, players, news, matches..."
                      className="flex-1 border-0 bg-transparent focus-visible:ring-0
text-lg placeholder:text-zinc-500"
                    />
                    <Button
                      variant="ghost"
                      size="icon"
                      onClick={() => setIsOpen(false)}
                      className="text-zinc-400 hover:text-white"
                    >
                      <X className="w-5 h-5" />
                    </Button>
                  </div>

                  {/* Results */}
                  <div className="max-h-96 overflow-y-auto">
                    {isLoading && (
                      <div className="p-8 text-center text-zinc-500">
                        Searching...
                      </div>
                    )}

                    {!isLoading && results.length === 0 && query.length >= 2 && (
                      <div className="p-8 text-center text-zinc-500">
                        No results found for "{query}"
                      </div>
                    )}
                    {!isLoading && results.length > 0 && (
                      <div className="divide-y divide-zinc-800">
                        {results.map((result) => (
                          <Link
                            key={`${result.type}-${result.id}`}
                            href={result.href}
                            onClick={() => setIsOpen(false)}
                            className="flex items-center gap-4 p-4 hover:bg-zinc-800/50 transition-colors"
                          >
                            {result.image ? (
                              <div className="w-10 h-10 rounded-lg overflow-hidden relative bg-zinc-800">
                                <Image
                                  src={result.image}
                                  alt={result.title}
                                  fill
                                  className="object-cover"
                                />
                              </div>
                            ) : (
                              <div className="w-10 h-10 rounded-lg bg-zinc-800 flex items-center justify-center">
                                {getIcon(result.type)}
                              </div>
                            )}
                            <div className="flex-1">
                              <p className="font-medium text-white">{result.title}</p>
                              {result.subtitle && (
                                <p className="text-sm text-zinc-400">{result.subtitle}</p>
                              )}
                            </div>
                            <span className="text-xs text-zinc-500 capitalize">{result.type}</span>
                          </Link>
                        ))}
                      </div>
                    )}
                  </div>

                  {/* Footer */}
                  <div className="p-3 border-t border-zinc-800 text-xs text-zinc-500 flex items-center justify-between">
                    <span>Type to search</span>
                    <span>Press ESC to close</span>
                  </div>
                </div>
              </motion.div>
            </>
          )}
        </AnimatePresence>
      </>
```

```
  );
}
```

Create `lib/hooks/useDebounce.ts`:

```
import { useState, useEffect } from 'react';

export function useDebounce<T>(value: T, delay: number): T {
  const [debouncedValue, setDebouncedValue] = useState<T>(value);

  useEffect(() => {
    const handler = setTimeout(() => {
      setDebouncedValue(value);
    }, delay);

    return () => {
      clearTimeout(handler);
    };
  }, [value, delay]);

  return debouncedValue;
}
```

Create `app/api/public/search/route.ts`:

```
import { createClient } from '@/lib/supabase/server';
import { NextRequest, NextResponse } from 'next/server';

export async function GET(request: NextRequest) {
  const query = request.nextUrl.searchParams.get('q');

  if (!query || query.length < 2) {
    return NextResponse.json([]);
  }

  try {
    const supabase = await createClient();
    const searchTerm = `%${query}%`;
    const results = [];

    // Search teams
    const { data: teams } = await supabase
      .from('teams')
      .select('id, name_en, name_am, slug, logo_url')
      .or(`name_en.ilike.${searchTerm},name_am.ilike.${searchTerm}`)
      .limit(5);

    if (teams) {
      results.push(...teams.map(t => ({
        type: 'team',
        id: t.id,
        title: t.name_en,
        subtitle: t.name_am,
        image: t.logo_url,
        href: `/teams/${t.slug}`,
      })));
    }

    // Search players
    const { data: players } = await supabase
      .from('players')
      .select('id, name_en, name_am, slug, photo_url, position_en')
      .or(`name_en.ilike.${searchTerm},name_am.ilike.${searchTerm}`)
      .limit(5);

    if (players) {
      results.push(...players.map(p => ({
        type: 'player',
        id: p.id,
        title: p.name_en,
        subtitle: p.position_en,
        image: p.photo_url,
        href: `/players/${p.slug}`,
      })));
    }

    // Search news
    const { data: news } = await supabase
      .from('news')
      .select('id, title_en, title_am, slug, thumbnail_url')
      .eq('is_published', true)
      .or(`title_en.ilike.${searchTerm},title_am.ilike.${searchTerm}`)
      .limit(5);

    if (news) {
      results.push(...news.map(n => ({
        type: 'news',
        id: n.id,
        title: n.title_en,
        subtitle: n.title_am,
        image: n.thumbnail_url,
        href: `/news/${n.slug}`,
      })));
    }

    // Search matches (by team names - more complex)
    const { data: matches } = await supabase
      .from('matches')
      .select(`
        id,
        date,
        home_team:teams!matches_home_team_id_fkey(name_en),
        away_team:teams!matches_away_team_id_fkey(name_en)
      `)
      .limit(5);

    if (matches) {
      const matchResults = matches
        .filter(m =>
          m.home_team?.name_en?.toLowerCase().includes(query.toLowerCase()) ||
          m.away_team?.name_en?.toLowerCase().includes(query.toLowerCase())
        )
        .slice(0, 5)
        .map(m => ({
          type: 'match',
          id: m.id,
          title: `${m.home_team?.name_en} vs ${m.away_team?.name_en}`,
          subtitle: new Date(m.date).toLocaleDateString(),
          href: `/matches/${m.id}`,
        }));
      results.push(...matchResults);
    }

    return NextResponse.json(results.slice(0, 15));
  } catch (error) {
    console.error('Search error:', error);
    return NextResponse.json({ error: 'Search failed' }, { status: 500 });
  }
}
```

PHASE 3: League Pages Season Integration

Task 3.1: Create SeasonToggle Component

Create `components/shared/SeasonToggle.tsx`:

```
"use client";

import { useState, useEffect } from "react";
import { useSeasons } from "@/lib/hooks/public/useSeasons";
import { ChevronDown, Calendar } from "lucide-react";
import { Button } from "@/components/ui/button";
import {
  DropdownMenu,
  DropdownMenuContent,
  DropdownMenuItem,
  DropdownMenuTrigger,
} from "@/components/ui/dropdown-menu";
import { cn } from "@/lib/utils";

interface SeasonToggleProps {
  currentSeasonId?: string;
  onSeasonChange: (seasonId: string) => void;
  className?: string;
}

export default function SeasonToggle({
  currentSeasonId,
  onSeasonChange,
  className,
}: SeasonToggleProps) {
  const { data: seasons, isLoading } = useSeasons();
  const [selectedSeason, setSelectedSeason] = useState<string | undefined>
(currentSeasonId);

  // Set default to current season
  useEffect(() => {
    if (seasons && !selectedSeason) {
      const current = seasons.find(s => s.is_current);
      if (current) {
        setSelectedSeason(current.id);
        onSeasonChange(current.id);
      }
    }
  }, [seasons, selectedSeason, onSeasonChange]);

  const handleChange = (seasonId: string) => {
    setSelectedSeason(seasonId);
    onSeasonChange(seasonId);
  };

  const currentSeasonName = seasons?.find(s => s.id === selectedSeason)?.name ||
"Select Season";

  if (isLoading) {
    return (
      <div className="h-12 w-40 bg-zinc-800 animate-pulse rounded-xl" />
    );
  }

  return (
    <DropdownMenu>
      <DropdownMenuTrigger asChild>
        <Button
          variant="outline"
          className={cn(
            "h-12 px-6 text-base font-semibold bg-zinc-900/80 border-primary/30
hover:border-primary hover:bg-zinc-800 transition-all",
            className
          )}
        >
          <Calendar className="w-5 h-5 mr-2 text-primary" />
          <span className="text-white">{currentSeasonName}</span>
          <ChevronDown className="w-4 h-4 ml-2 text-zinc-400" />
        </Button>
      </DropdownMenuTrigger>
      <DropdownMenuContent
```

```
        align="start"
        className="w-48 bg-zinc-900 border-zinc-800"
      >
        {seasons?.map((season) => (
          <DropdownMenuItem
            key={season.id}
            onClick={() => handleChange(season.id)}
            className={cn(
              "cursor-pointer",
              selectedSeason === season.id && "bg-primary/10 text-primary"
            )}
          >
            <span className="flex items-center gap-2">
              {season.name}
              {season.is_current && (
                <span className="text-xs bg-primary/20 text-primary px-2 py-0.5
rounded-full">
                  Current
                </span>
              )}
            </span>
          </DropdownMenuItem>
        ))}
      </DropdownMenuContent>
    </DropdownMenu>
  );
}
```

Task 3.2: Update League Page Header

Modify `components/premier-league/PremierLeagueHeader.tsx` (and similar for other league headers):

Add SeasonToggle to the header section.

PHASE 4: Match Control Panel (Continue in next workflow file or chat)

This section is extensive and will be implemented in the next phase.

Key tasks:

- Time persistence implementation
- Pause vs postpone separation
- Event editing/deletion
- Penalty shootout panel
- Restart functionality
- Auto-timeout

📝 Progress Tracking

Use this checklist to track progress across chat sessions:

Phase 1: Infrastructure

- [ ] Database migration run
- [ ] TypeScript types regenerated
- [ ] Season hooks created
- [ ] Season API routes created

Phase 2: Main Site UI

- [ ] Navbar enlarged
- [ ] Women's League moved to dropdown
- [ ] Features menu added
- [ ] AdBanner component updated
- [ ] Global search implemented
- [ ] OtherNews component created
- [ ] Standings limit removed
- [ ] Ad placements added to home page

Phase 3: League Pages

- [ ] SeasonToggle component created
- [ ] League headers updated
- [ ] Season filtering implemented
- [ ] Standings color coding fixed
- [ ] Team row click handlers added

Phase 4: Match Control Panel

- [ ] Time persistence implemented
- [ ] Status handling updated
- [ ] Pause/resume working
- [ ] Penalty management working
- [ ] Event editing implemented
- [ ] Restart functionality added
- [ ] Auto-timeout implemented

Phase 5: CMS Modules

- [ ] Season management pages created
- [ ] Dashboard upgraded
- [ ] Ad management pages created
- [ ] Cup management pages created

Phase 6: Cup System

- [ ] Cup pages created
- [ ] Bracket visualization working
- [ ] Group stage display working

Phase 7: Polish

- [ ] All automation tested
- [ ] Stats displaying correctly
- [ ] Final bug fixes complete

🔄 Cross-Chat Handoff Template

When starting a new chat, copy and paste this:

```
I'm continuing Pana Sports v2.0 development.

Reference: /.agent/workflows/v2-architecture.md

Current Phase: [PHASE NUMBER]
Last Completed: [LAST TASK]
Next Task: [NEXT TASK]

Any blockers: [YES/NO - details if yes]
```