WORK 12 – Admin Login Redirect & Access Fix - COMPLETE

Date: November 26, 2024
Status: ✅ Fixed & Verified
Deployment: https://catchbarrels.app

🎯 Problem Summary

Admin users could authenticate successfully but were not properly redirected to the `/admin` dashboard. The issues identified were:

1. **Redirect Callback Issue**: The NextAuth `redirect` callback always sent users to `/dashboard`, ignoring the `callbackUrl` parameter
2. **Client-Side Navigation Issue**: Using `router.refresh()` after `router.push()` was causing navigation problems
3. **Middleware Product Gating**: Admins might be blocked by product ownership checks on non-admin routes
4. **No Role-Based Redirect Helper**: No centralized function to determine where users should go based on role

---

✅ What Was Fixed

1. Redirect Callback in `lib/auth-options.ts`

Before:

```
async redirect({ url, baseUrl }) {
  // Default redirect to dashboard
  if (url === baseUrl || url === `${baseUrl}/` || url.includes('/auth/login')) {
    return `${baseUrl}/dashboard`;  // ❌ Always goes to dashboard
  }
  // ...
}
```

After:

```
async redirect({ url, baseUrl }) {
  // Handle callback URLs properly
  try {
    // If URL is the base URL or login page without a callbackUrl, go to dashboard
    if (url === baseUrl || url === `${baseUrl}/`) {
      return `${baseUrl}/dashboard`;
    }

    // If it's the login page with a callbackUrl parameter, extract and use it
    if (url.includes('/auth/login') || url.includes('/auth/admin-login')) {
      const urlObj = new URL(url);
      const callbackUrl = urlObj.searchParams.get('callbackUrl');

      if (callbackUrl) {
        // If callback URL is relative, prepend baseUrl
        if (callbackUrl.startsWith('/')) {
          return `${baseUrl}${callbackUrl}`;  // ✅ Respects callbackUrl
        }
        // If callback URL is on same origin, use it
        if (callbackUrl.startsWith(baseUrl)) {
          return callbackUrl;
        }
      }

      // No callback URL, default to dashboard
      return `${baseUrl}/dashboard`;
    }

    // Allow relative callback URLs
    if (url.startsWith('/')) {
      return `${baseUrl}${url}`;
    }

    // Allow callback URLs on the same origin
    const urlObj = new URL(url);
    if (urlObj.origin === baseUrl) {
      return url;
    }

    // Default fallback
    return `${baseUrl}/dashboard`;
  } catch (error) {
    console.error('Redirect error:', error);
    return `${baseUrl}/dashboard`;
  }
}
```

Impact: Admin login now properly respects the `callbackUrl=/admin` parameter set by the login form.

---

2. Login Client Navigation in `app/auth/login/login-client.tsx`

Before:

```
else if (result?.ok) {
  const welcomeMsg = loginMode === 'admin' ? 'Welcome, Admin!' : 'Welcome back!';
  toast.success(welcomeMsg, {
    description: 'Redirecting...',
  });

  // Small delay to let session update
  setTimeout(() => {
    router.push(callbackUrl);
    router.refresh();  // ❌ Causes issues with NextAuth
  }, 100);
}
```

After:

```
else if (result?.ok) {
  const welcomeMsg = loginMode === 'admin' ? 'Welcome, Admin!' : 'Welcome back!';
  toast.success(welcomeMsg, {
    description: 'Redirecting...',
  });

  // Navigate to callback URL (no router.refresh() needed with NextAuth)
  router.push(callbackUrl);  // ✅ Clean navigation
}
```

Impact: Removed problematic `router.refresh()` call that could interfere with NextAuth's session handling.

---

3. Middleware Admin Bypass in `middleware.ts`

Before:

```
// Check for admin/coach-only routes (/admin)
if (pathname.startsWith('/admin')) {
  const userRole = (token as any).role || 'player';
  const hasAdminAccess = userRole === 'admin' || userRole === 'coach';

  if (!hasAdminAccess) {
    // Not authorized - redirect to dashboard with error message
    const dashboardUrl = new URL('/dashboard', request.url);
    dashboardUrl.searchParams.set('error', 'unauthorized');
    return NextResponse.redirect(dashboardUrl);
  }

  // Has admin/coach role - allow access to admin routes
  return NextResponse.next();
}

// User is authenticated - check product ownership for player routes
const membershipTier = (token as any).membershipTier || 'free';
const membershipStatus = (token as any).membershipStatus || 'inactive';

// Check if user has an active paid membership
const hasProduct =
  membershipStatus === 'active' &&
  membershipTier !== 'free';

// If no product, redirect to purchase page
if (!hasProduct) {
  // ... redirect logic
  // ❌ Could block admin if they try to access /dashboard
}
```

After:

```
// Check if user is admin/coach (they get full access to everything)
const userRole = (token as any).role || 'player';
const isAdmin = userRole === 'admin' || userRole === 'coach';

// Admin/Coach access control
if (pathname.startsWith('/admin')) {
  if (!isAdmin) {
    // Not authorized for admin area - redirect to dashboard with error message
    const dashboardUrl = new URL('/dashboard', request.url);
    dashboardUrl.searchParams.set('error', 'unauthorized');
    return NextResponse.redirect(dashboardUrl);
  }

  // Has admin/coach role - allow access to admin routes
  return NextResponse.next();
}

// Admins/coaches bypass product gating - they have full access
if (isAdmin) {
  return NextResponse.next();  // ✅ Full app access for admins
}

// Regular user - check product ownership for player routes
// ... existing product gating logic
```

Impact: Admins now have full access to the entire app (both `/admin` and athlete routes) without product gating checks.

---

4. Role Utilities Helper in `lib/role-utils.ts` (NEW FILE)

```
import { Session } from 'next-auth';
import { JWT } from 'next-auth/jwt';

/**
 * Check if a user has admin or coach role
 * @param sessionOrToken - NextAuth session or JWT token
 * @returns true if user is admin or coach
 */
export function isAdmin(sessionOrToken: Session | JWT | null): boolean {
  if (!sessionOrToken) return false;

  // Check if it's a session object
  if ('user' in sessionOrToken) {
    const user = sessionOrToken.user as any;
    return user?.role === 'admin' || user?.role === 'coach' || user?.isCoach === true;
  }

  // Check if it's a token object
  const token = sessionOrToken as any;
  return token?.role === 'admin' || token?.role === 'coach' || token?.isCoach ===
true;
}

/**
 * Get the appropriate redirect URL based on user role
 * @param sessionOrToken - NextAuth session or JWT token
 * @param fallback - Fallback URL if no role-based redirect
 * @returns Redirect URL
 */
export function getRoleBasedRedirect(sessionOrToken: Session | JWT | null, fallback: s
tring = '/dashboard'): string {
  if (isAdmin(sessionOrToken)) {
    return '/admin';
  }
  return fallback;
}
```

Impact: Provides reusable helper functions for consistent role checking across the app.

🧪 Testing Results

Manual Testing Performed

✅ Admin Login Flow

1. Navigate to `/auth/login` ✅
2. Click “Admin” tab ✅
3. Enter admin credentials (`coach@catchbarrels.app` / `CoachBarrels2024!`) ✅
4. Click “Admin Sign In” ✅
5. **Verify redirect to `/admin`** ✅
6. **Verify Coach Control Room dashboard loads** ✅
7. **Verify “MOMENTUM 84” badge displays** ✅
8. **Verify dashboard metrics display (0 athletes, 0 active, 0 avg score)** ✅

✅ Admin Full Access

1. Log in as admin ✅
2. Access `/admin` ✅ (loads Coach Control Room)
3. Access `/dashboard` ✅ (admin can view athlete dashboard)
4. Access `/video` ✅ (admin can view video section)
5. No product gating blocks admin ✅

✅ Session Verification

1. Used debug endpoint `/api/debug-session` to verify:
   - `session.user.role === 'admin'` ✅
   - `session.user.isCoach === true` ✅
   - `session.user.membershipTier === 'elite'` ✅
   - `session.user.membershipStatus === 'active'` ✅

✅ Logout & Re-login

1. Click logout ✅
2. Redirect to `/auth/login` ✅
3. Re-login as admin ✅
4. Redirect to `/admin` ✅

📝 Files Modified

Updated Files:

```
lib/auth-options.ts                    - Fixed redirect callback
app/auth/login/login-client.tsx        - Removed router.refresh()
middleware.ts                          - Added admin bypass for product gating
```

New Files:

```
lib/role-utils.ts                      - Role checking helper functions
docs/WORK12_ADMIN_LOGIN_REDIRECT_FIX_COMPLETE.md - This documentation
```

Temporary Files (Removed):

```
app/api/debug-session/route.ts         - Debug endpoint (removed for security)
```

🔒 Security Verification

✅ Route Protection

- `/admin` routes require `role === 'admin'` or `role === 'coach'` ✅
- Non-admin users redirected to `/dashboard` with error message ✅
- Server-side validation in both middleware AND page components ✅

✅ Session Security

- JWT tokens include role information ✅
- Session includes role and coach status ✅
- No client-side role checks that can be bypassed ✅

✅ Admin Privileges

- Admins bypass product gating ✅
- Admins have full app access ✅
- Admin seed script sets proper membership tier (elite) ✅

🚀 Deployment Status

Production Deployment

- **URL**: https://catchbarrels.app
- **Login Page**: https://catchbarrels.app/auth/login
- **Admin Dashboard**: https://catchbarrels.app/admin
- **Deployment Date**: November 26, 2024
- **Build Status**: ✅ Success
- **Runtime Status**: ✅ Live
- **Verified**: ✅ Admin login and redirect working

📊 Admin Account Details

Production Admin Account

Email: `coach@catchbarrels.app`
Password: `CoachBarrels2024!`
Role: `admin`
Is Coach: `true`
Membership: `elite`
Status: ✅ Active

Login URL: https://catchbarrels.app/auth/login → Admin tab

🔍 Root Cause Analysis

Why Did It Fail Before?

1. **Redirect Callback**:
   - The callback was checking if the URL “includes ‘/auth/login’“
   - But it was blindly returning `/dashboard` without extracting the `callbackUrl` parameter
   - Result: All logins went to `/dashboard`, ignoring role

2. **Client-Side Navigation**:
   - Using `router.refresh()` after `router.push()` was causing the navigation to be interrupted
   - NextAuth manages its own session refresh, so manual refresh was conflicting

3. **Middleware Logic**:
   - Admin routes were checked first (correct)
   - But then product gating was applied to ALL authenticated users
   - Admins trying to access `/dashboard` would be blocked if they didn't pass product checks
   - Solution: Add early return for admins BEFORE product gating checks

---

✨ What Works Now

Admin Login Flow

```
1. User visits /auth/login
   ↓
2. Clicks "Admin" tab
   ↓
3. Enters admin credentials
   ↓
4. Client sets callbackUrl=/admin
   ↓
5. signIn('admin-credentials', { callbackUrl: '/admin' })
   ↓
6. NextAuth validates credentials
   ↓
7. JWT created with role='admin'
   ↓
8. Redirect callback extracts callbackUrl parameter
   ↓
9. Returns /admin
   ↓
10. Client navigates to /admin
   ↓
11. Middleware checks role (admin) → allows
   ↓
12. Page component checks role (admin) → renders
   ↓
13. ✅ Coach Control Room displays
```

Admin Full Access

```
Admin authenticated
   ↓
Middleware checks token.role
   ↓
Is admin? YES
   ↓
Bypass all product gating
   ↓
Allow access to:
   • /admin (Coach Control Room)
   • /dashboard (Athlete Dashboard)
   • /video (Video Library)
   • /sessions (Session History)
   • ALL routes in the app
   ↓
✅ Full app access granted
```

🎯 Success Metrics

Technical

- ✅ Zero build errors
- ✅ Zero TypeScript errors
- ✅ All routes protected correctly
- ✅ Admin redirect working
- ✅ Session includes role data
- ✅ Middleware enforces access control

Security

- ✅ Server-side role validation
- ✅ JWT token security
- ✅ Double-layer protection (middleware + page)
- ✅ Admin bypass properly scoped

User Experience

- ✅ Clean admin login flow
- ✅ Proper redirect to /admin
- ✅ Full app access for admins
- ✅ No product gating interference
- ✅ Smooth navigation (no refresh issues)

🔮 Future Enhancements

1. **Role-Based Permissions**:
   - Granular permissions within admin role
   - Different access levels for “admin” vs “coach”

2. **Admin User Management**:
   - UI to create/edit/delete admin users
   - Role assignment interface

3. **Audit Logging**:
   - Log all admin actions
   - Track login history

4. **MFA for Admins**:
   - Two-factor authentication for admin accounts
   - SMS or authenticator app support

📚 Related Documentation

- **Admin Setup**: `/docs/ADMIN_SETUP.md`
- **Admin Login Flow**: `/docs/ADMIN_LOGIN_FLOW.md`
- **Admin Implementation**: `/docs/ADMIN_IMPLEMENTATION_COMPLETE.md`
- **This Fix**: `/docs/WORK12_ADMIN_LOGIN_REDIRECT_FIX_COMPLETE.md`

🎉 Summary

Problem: Admin login worked, but redirect to `/admin` failed
Root Cause: Redirect callback ignored `callbackUrl` parameter, always sent to `/dashboard`
Solution: Fixed redirect callback to extract and respect `callbackUrl`, removed problematic `router.refresh()`, added admin bypass in middleware
Result: ✅ Admin login now redirects to `/admin` Coach Control Room successfully
Status: 🚀 **Production Ready & Verified**

Last Updated: November 26, 2024
Version: 1.1
Verified By: DeepAgent
Status: ✅ **FIXED & DEPLOYED**