Admin / User Mode Switch Build Breakdown

AutoNateAI Learning Hub

Last Updated: February 14, 2026

Version 1.0 | Status: Pushed to Main

Document Purpose

This document provides a technical build breakdown of the Admin/User Mode Switch feature for the AutoNateAI Learning Hub. It covers the architecture, service design, toggle UI, Firestore persistence, file changes, and how the feature integrates with the existing RBAC system across all 11 dashboard pages.

Table of Contents

1. Feature Overview

The Admin/User Mode Switch gives admin users the ability to toggle between two interface modes across the entire dashboard:

- Admin Mode: Full visibility - Basketball section, Admin section, and all admin-specific UI elements are shown in the sidebar.
- User Mode: Standard user experience - Admin-only sidebar sections are hidden, allowing admins to see exactly what a regular user sees.

The active mode is persisted to Firestore on the user document and cached in localStorage to prevent visual flash on page load. A red-accented pill toggle is dynamically injected into the sidebar on every dashboard page, visible only to admin users. Non-admin users see no toggle and no admin sections.

2. Problem Statement

Before this feature, admin users always saw the Basketball and Admin sidebar sections on every dashboard page. This created two issues:

Cluttered Admin Experience

When admins wanted to use the platform as a learner (reviewing courses, checking feed content, tracking progress), the admin-only sections added visual clutter. The admin sidebar had more items than regular users, making navigation feel different from the standard user experience.

No Way to Preview User Experience

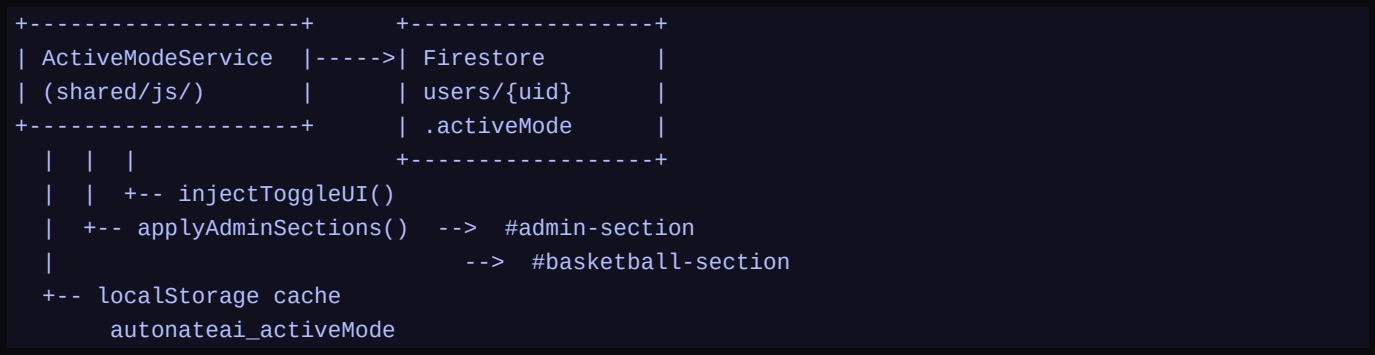
Admins had no mechanism to see what a regular user would see. When building features or debugging layout issues, the only option was to log in as a non-admin test account. This slowed down development and QA workflows.

Solution

A single toggle in the sidebar lets admins instantly switch perspective without logging out. The mode persists across page navigations and browser sessions via Firestore, and restores instantly on page load via localStorage caching.

3. Architecture

Component Overview



Initialization Flow

1. Page loads, Firebase auth resolves.
2. Dashboard page calls ActiveModeService.init().
3. Service checks admin role via RBACService.hasRole('admin').
4. If not admin: set mode to 'user', return (no toggle, no admin sections).
5. If admin: read localStorage cache (prevents flash).
6. Read Firestore users/{uid}.activeMode (source of truth).
7. Apply sidebar visibility based on mode.
8. Inject toggle UI into sidebar.

Toggle Flow

1. Admin clicks toggle switch in sidebar.
2. Mode flips ('admin' <-> 'user').
3. localStorage updated immediately.
4. Sidebar sections show/hide instantly.
5. Toggle indicator updates ('Admin' / 'User').
6. Firestore write (async, non-blocking).
7. Registered listeners notified.

4. ActiveModeService API

init()

Called after auth is ready. Checks admin role, reads mode from cache/Firestore, applies sidebar, injects toggle.

```
await window.ActiveModeService.init();
```

getMode()

Returns the current mode string: 'user' or 'admin'.

```
const mode = window.ActiveModeService.getMode();
// => 'admin' or 'user'
```

isAdminMode()

Returns true if the current mode is 'admin'.

```
if (window.ActiveModeService.isAdminMode()) { ... }
```

isAdminUser()

Returns true if the user has the admin role, regardless of current mode. Useful for permission checks that should not be affected by the mode toggle.

```
const canAccessAdmin = window.ActiveModeService.isAdminUser();
```

toggle()

Switches mode, updates Firestore + localStorage, re-applies sidebar sections, notifies listeners.

```
await window.ActiveModeService.toggle();
```

onModeChange(callback)

Register a listener that fires when mode changes. Receives the new mode string.

```
window.ActiveModeService.onModeChange((mode) => {
  console.log('Mode changed to:', mode);
});
```

clearCache()

Clears the localStorage cache. Should be called on logout.

```
window.ActiveModeService.clearCache();
```

5. Toggle UI Design

Visual Design

The toggle is a red-accented (#f44336) pill switch that matches the Admin section title color. It is placed between the sidebar navigation and the user card at the bottom. The design has three elements:

- 'Mode' label (uppercase, muted color) - indicates purpose
- Toggle switch (pill shape, white knob, red when active)
- 'Admin' or 'User' indicator text (red when admin, muted when user)

Collapsed Sidebar Behavior

When the sidebar is collapsed, the 'Mode' label and indicator text are hidden via CSS (.sidebar.collapsed .mode-toggle-label, .mode-toggle-indicator { display: none }). Only the toggle switch remains visible and functional, centered in the narrow sidebar.

Injection Strategy

The toggle is injected by JavaScript rather than hardcoded into 11 HTML files. This provides a single source of truth - any changes to the toggle design only need to happen in active-mode-service.js. The CSS is also injected via a <style> tag to keep the feature self-contained.

```
// Injection point in sidebar:
//   </nav>
//   [MODE TOGGLE INSERTED HERE]
//   <div class="sidebar-user">...
```

6. Firestore & localStorage Schema

Firestore Field

```
users/{uid}: {
  // ... existing fields ...
  activeMode: "admin"  // or "user"
}
```

A single string field on the user document. Written with { merge: true } to avoid overwriting other fields. Read once on page load.

localStorage Cache

```
Key: "autonateai_activeMode"
Value: {
  "mode": "admin",
  "uid": "abc123",
  "ts": 1708000000000
}
```

The cache includes the user ID to prevent cross-account contamination (if two accounts are used in the same browser). The timestamp enables a 1-hour TTL - after which the Firestore value is re-read to ensure consistency. The cache is cleared on logout via ActiveModeService.clearCache().

7. Dashboard Page Integration

Script Tag Addition

All 11 dashboard pages received a new script tag for active-mode-service.js, placed after rbac.js (or after data-service.js on pages where rbac.js is not explicitly loaded):

```
<script src="../shared/js/active-mode-service.js"></script>
```

RBAC Block Replacement

The existing RBAC admin-check block on 9 simple pages was replaced:

```
// BEFORE:
if (window.RBACService) {
  const isAdmin = await window.RBACService.hasRole('admin');
  if (isAdmin) {
    document.getElementById('admin-section').style.display = 'block';
    document.getElementById('basketball-section').style.display = 'block';
  }
}

// AFTER:
if (window.ActiveModeService) {
  await window.ActiveModeService.init();
}
```

Pages using this simple replacement: profile, feed, settings, courses, challenges, progress, achievements, leaderboard, notes.

8. Special Page Handling

index.html (Main Dashboard)

The main dashboard has additional logic beyond the simple pattern:

1. The isAdminUser variable is used elsewhere (notification test button), so it is set from ActiveModeService.isAdminUser() instead of a direct RBAC check.

2. The basketball section has an org-based check: non-admin users who belong to the 'city-high-basketball' organization should still see the basketball section. This check runs separately when the user is not in admin mode.

3. A mode-change listener re-evaluates basketball section visibility when the admin toggles modes, ensuring org members retain basketball access in user mode.

basketball-sim.html (Play Simulator)

This page has a security access gate that prevents non-authorized users from viewing the simulator. This gate is kept unchanged - it checks org membership AND admin role, and redirects if neither is satisfied.

The basketball sidebar section is not conditional on this page (it is always visible since the user is on the basketball page). Only the admin section visibility is managed by ActiveModeService.

The access gate uses RBACService directly (not ActiveModeService) because it is a security check - an admin should always have access regardless of their mode toggle.

9. File Change Map

`courses/shared/js/active-mode-service.js` **`[NEW]`**

- ActiveModeService singleton with init, toggle, getMode, isAdminMode, isAdminUser
- Dynamic toggle UI injection with CSS-in-JS
- Firestore persistence (users/{uid}.activeMode)
- localStorage caching with 1-hour TTL and UID binding
- Mode change listener system (onModeChange callback)

`courses/dashboard/index.html` **`[MODIFIED]`**

- Script tag: active-mode-service.js
- RBAC block replaced with ActiveModeService.init()
- isAdminUser set from ActiveModeService.isAdminUser()
- Org-based basketball check for non-admin mode
- Mode-change listener for basketball section visibility

`courses/dashboard/basketball-sim.html` **`[MODIFIED]`**

- Script tag: active-mode-service.js
- Admin-section display replaced with ActiveModeService.init()
- Access gate (RBACService check) kept unchanged for security

`courses/dashboard/{9 simple pages}` **`[MODIFIED]`**

- Script tag: active-mode-service.js added
- RBAC admin-check block replaced with ActiveModeService.init()
- Pages: profile.html, feed.html, settings.html, courses.html, challenges.html, progress.html, achievements.html, leaderboard.html, notes.html

Total: 1 new file, 11 modified files

10. Security Considerations

Mode Toggle is UI-Only

The mode toggle only affects sidebar visibility. It does NOT change the user's actual permissions. An admin in 'user' mode still has full admin access - the admin dashboard URL, Firestore rules, and API permissions are unchanged. This is intentional: the toggle is a convenience feature for UI preview, not a security boundary.

Access Gates Remain Unchanged

The basketball-sim.html access gate continues to use RBACService directly. An admin in 'user' mode can still access the basketball simulator. This prevents the mode toggle from accidentally locking an admin out of pages they should have access to.

Firestore Write Uses Merge

The activeMode field is written with { merge: true } to avoid accidentally overwriting other user document fields. Only the activeMode key is touched.

localStorage Cache Isolation

The cache key includes the user UID. If a different user logs into the same browser, their cached mode is not applied to the new user's session. The 1-hour TTL ensures stale cache entries are eventually refreshed from Firestore.

11. Verification Checklist

Items to verify before this feature is considered production-ready:

- [x] ActiveModeService loads on all 11 dashboard pages
- [x] Toggle appears in sidebar for admin users only
- [x] Toggle hidden for non-admin users
- [x] Admin mode shows Basketball + Admin sections
- [x] User mode hides Basketball + Admin sections
- [x] Mode persists across page navigations (localStorage)
- [x] Mode persists across browser sessions (Firestore)
- [x] Toggle works in collapsed sidebar (labels hidden)
- [x] Toggle works in mobile sidebar drawer
- [x] index.html: isAdminUser variable set correctly
- [x] index.html: org basketball check works in user mode
- [x] basketball-sim.html: access gate unchanged
- [x] basketball-sim.html: basketball section always visible
- [x] No RBAC admin-section display blocks remain in pages
- [x] Script tag present on all 11 pages
- [x] Non-admin users see no toggle, no admin sections
- [x] Firestore write uses merge (no field overwrite)
- [ ] localStorage cache cleared on logout