🎯 RBAC Implementation Progress Tracker

Project: Payroll SaaS
Branch: refactor/rbac-dynamic
Last Updated: November 15, 2025
Objective: Transform the application into a dynamic RBAC system with clean, scalable architecture

---

📊 Overall Progress Summary

Phase	Status	Progress	Priority
Phase 1: Architecture Restructuring	🔴 Not Started	0/10	⚠️ Critical
Phase 2: Database & Models	✅ Completed	4/4	High
Phase 3: RBAC Core System	✅ Completed	6/6	High
Phase 4: Middleware & Guards	🟡 Partial	2/5	High
Phase 5: UI Components	🔴 Not Started	0/8	Medium
Phase 6: Contracts System	🔴 Not Started	0/6	Medium
Phase 7: Testing & Validation	🔴 Not Started	0/5	Medium
Phase 8: Documentation	🔴 Not Started	0/3	Low
Phase 9: Performance Optimization	🔴 Not Started	0/2	Low
Phase 10: Final Cleanup	🔴 Not Started	0/1	Low

Total Progress: 12/50 steps completed (24%)

📋 Phase 1: Architecture Restructuring & Cleanup (0/10) 🔴

Objective

Refactor from role-based structure to domain-based (modules) structure with dynamic routing.

Steps

- [ ] **Step 1:** Create new folder structure
- **Status:** ❌ Not Started
- **Description:** Migrate from role-based folders to modules structure
- **Files:** `app/(dashboard)/(modules)/`
- **Notes:** Current structure uses (modules) route groups but needs complete reorganization
- [ ] **Step 2:** Create dynamic routing system
- **Status:** ⚠️ Partial (routing config exists but not fully used)
- **Description:** Implement `lib/routing/dynamic-router.ts`
- **Files:** Need to create dynamic-router.ts
- **Dependencies:** server/rbac/permissions.ts
- [ ] **Step 3:** Create usePermissions hook
- **Status:** ✅ **DONE** (Exists in hooks/use-permissions)
- **Description:** Hook for client-side permission checks
- **Files:** `lib/hooks/usePermissions.ts` or `hooks/use-permissions.ts`
- [ ] **Step 4:** Create PermissionGuard component
- **Status:** ❌ Not Started
- **Description:** Wrapper component to show/hide UI elements based on permissions
- **Files:** Need `components/guards/permission-guard.tsx`
- **Dependencies:** usePermissions hook
- [ ] **Step 5:** Create ActionButton with permissions
- **Status:** ❌ Not Started
- **Description:** Reusable button with automatic permission handling
- **Files:** Need `components/ui/action-button.tsx`
- **Dependencies:** PermissionGuard
- [ ] **Step 6:** Create dynamic menu system
- **Status:** ✅ **DONE** (lib/dynamicMenuConfig.ts exists)
- **Description:** Menu configuration based on permissions
- **Files:** `lib/dynamicMenuConfig.ts` ✅ (FIXED: removed /modules/ prefix)
- [ ] **Step 7:** Refactor Sidebar
- **Status:** ✅ **DONE** (uses dynamicMenuConfig)
- **Description:** Sidebar uses dynamic menu config

- **Files:** `components/layout/sidebar.tsx` ✅ (FIXED: routing issue)
- [ ] **Step 8:** Create dashboard with intelligent routing
- **Status:** ✅ **DONE** (NEW: Dynamic dashboard with tRPC stats)
- **Description:** Dashboard redirects based on permissions
- **Files:** `app/(dashboard)/dashboard/page.tsx` ✅ (ENHANCED with real data)
- [ ] **Step 9:** Update middleware
- **Status:** ⚠️ Needs Review
- **Description:** Middleware uses dynamic routing
- **Files:** `middleware.ts`
- **Dependencies:** dynamic-router.ts
- [ ] **Step 10:** Remove old menuConfig
- **Status:** ❌ Not Started
- **Description:** Clean up old role-based menu system
- **Files:** Check for `lib/menuConfig.ts` (old)

📋 Phase 2: Database & Models (4/4) ✅

Objective

Enhance Prisma schema with complete contract workflow and optimizations.

Steps

- [x] **Step 11:** Enhance Contract schema
- **Status:** ✅ Completed
- **Description:** Add workflow fields, documents, status history, notifications
- **Files:** `prisma/schema.prisma`
- **Note:** Schema has Contract model with relations
- [x] **Step 12:** Add performance indexes
- **Status:** ✅ Completed
- **Description:** Index frequently queried columns
- **Files:** `prisma/schema.prisma`
- [x] **Step 13:** Create TypeScript types
- **Status:** ✅ Completed
- **Description:** Contract types and enums
- **Files:** `lib/types/contracts.ts`
- [x] **Step 14:** Update lib/types.ts
- **Status:** ✅ Completed
- **Description:** Add contract entity types to audit system

- **Files:** `lib/types.ts`

📋 Phase 3: RBAC Core System (6/6) ✅

Objective

Build the core dynamic RBAC system with permissions and roles.

Steps

- [x] **Step 15:** Enhance permissions tree
- **Status:** ✅ Completed
- **Description:** Add granular permissions for contracts, workflows, documents
- **Files:** `server/rbac/permissions.ts`
- [x] **Step 16:** Create permission validator
- **Status:** ✅ Completed
- **Description:** Server-side permission validation utilities
- **Files:** `server/rbac/permission-validator.ts`
- [x] **Step 17:** Update permission seed
- **Status:** ✅ Completed
- **Description:** Add new permissions to database seed
- **Files:** `scripts/seed/00-permissions.ts`
- [x] **Step 18:** Improve role seed
- **Status:** ✅ Completed
- **Description:** Default roles with precise permissions
- **Files:** `scripts/seed/01-roles.ts`
- [x] **Step 19:** Create roleManagement router
- **Status:** ✅ Completed
- **Description:** tRPC CRUD for roles with permission assignment
- **Files:** `server/api/routers/roleManagement.ts`
- [x] **Step 20:** Register roleManagement router
- **Status:** ✅ Completed
- **Description:** Add to root router
- **Files:** `server/api/root.ts`

📋 Phase 4: Middleware & Guards (2/5) 🟡

Objective

Create middleware and guards for ownership and route protection.

Steps

- [ ] **Step 21:** Create ownership middleware
- **Status:** ⚠️ Needs Implementation
- **Description:** Ensure users access only their resources
- **Files:** Need `server/api/middleware/ownership.ts`
- [ ] **Step 22:** Create RouteGuard component
- **Status:** ❌ Not Started
- **Description:** Client-side route protection
- **Files:** Need `components/guards/route-guard.tsx`
- [x] **Step 23:** Create dashboard statistics router
- **Status:** ✅ **COMPLETED** (NEW)
- **Description:** tRPC router for dashboard stats
- **Files:** `server/api/routers/dashboard.ts` ✅
- [x] **Step 24:** Register dashboard router
- **Status:** ✅ **COMPLETED** (NEW)
- **Description:** Add to root router
- **Files:** `server/api/root.ts` ✅
- [ ] **Step 25:** Test permission enforcement
- **Status:** ❌ Not Started
- **Description:** Verify permissions work end-to-end

📋 Phase 5: UI Components (0/8) 🔴

Objective

Build reusable UI components with permission integration.

Steps

- [ ] **Step 26:** Create DataTable with permissions
- [ ] **Step 27:** Create Form components with permission validation
- [ ] **Step 28:** Create Modal/Dialog with guards
- [ ] **Step 29:** Create Tabs component with permission-based visibility
- [ ] **Step 30:** Create DropdownMenu with action permissions
- [ ] **Step 31:** Create Card components with permission checks
- [ ] **Step 32:** Create Badge/Status indicators
- [ ] **Step 33:** Create notification system

Status: All steps not started

📋 Phase 6: Contracts System (0/6) 🔴

Objective

Complete contract workflow with document management and signatures.

Steps

- [ ] **Step 34:** Create contract tRPC router
- [ ] **Step 35:** Implement contract workflow transitions
- [ ] **Step 36:** Build contract UI pages
- [ ] **Step 37:** Add document upload/management
- [ ] **Step 38:** Implement signature system
- [ ] **Step 39:** Create contract notifications

Status: All steps not started

📋 Phase 7: Testing & Validation (0/5) 🔴

Objective

Comprehensive testing of RBAC system.

Steps

- [ ] **Step 40:** Unit tests for permission validators
- [ ] **Step 41:** Integration tests for tRPC routers
- [ ] **Step 42:** E2E tests for permission flows
- [ ] **Step 43:** Test role-based access scenarios
- [ ] **Step 44:** Security audit

Status: All steps not started

📋 Phase 8: Documentation (0/3) 🔴

Objective

Document the RBAC system and architecture.

Steps

- [ ] **Step 45:** Create RBAC architecture documentation
- [ ] **Step 46:** Document permission tree structure
- [ ] **Step 47:** Create developer guide

Status: All steps not started

📋 Phase 9: Performance Optimization (0/2) 🔴

Objective

Optimize database queries and caching.

Steps

- [ ] **Step 48:** Implement query optimization
- [ ] **Step 49:** Add caching layer

Status: All steps not started

📋 Phase 10: Final Cleanup (0/1) 🔴

Objective

Remove deprecated code and finalize migration.

Steps

- [ ] **Step 50:** Remove all deprecated files and old code

Status: Not started

🎯 Current Status & Recent Changes

✅ Recently Completed (Nov 15, 2025)

1. **Fixed Sidebar Routing Issue** ✅
   - Removed incorrect `/modules/` prefix from all menu links in `lib/dynamicMenuConfig.ts`
   - Fixed links in `app/(dashboard)/(home)/page.tsx`
   - **Reason:** NextJS route groups `(modules)` don't appear in URLs

2. **Created Dynamic Dashboard with Real Data** ✅
   - New `server/api/routers/dashboard.ts` with statistics queries
   - Enhanced `app/(dashboard)/dashboard/page.tsx` with:
     - Real-time stats for contractors, contracts, invoices, agencies, etc.
     - Revenue tracking from paid invoices
     - Recent activity feed
     - Upcoming contract expirations
     - Role-based data display
     - Registered dashboard router in `server/api/root.ts`

3. **Improved Dashboard UX** ✅
   - Clickable stat cards linking to relevant pages
   - Color-coded status indicators
   - Loading states
   - Permission-based visibility
   - Professional layout with cards and badges

🚧 In Progress

1. **Dynamic Routing System**
   - Need to create `lib/routing/dynamic-router.ts`
   - Update middleware to use dynamic routing

2. **Permission Guards**
   - Need PermissionGuard component
   - Need RouteGuard component
   - Need ActionButton component

⚠️ Blockers & Issues

1. **Architecture Cleanup**
   - Phase 1 needs significant work
   - Old role-based structure mixed with new module structure

2. **Missing Components**
   - Permission guards not implemented
   - Dynamic router utilities missing

📌 Next Priority Steps

1. **Immediate (Must Do)**
   - [ ] Create `lib/routing/dynamic-router.ts`
   - [ ] Create `components/guards/permission-guard.tsx`
   - [ ] Create `components/guards/route-guard.tsx`
   - [ ] Update middleware to use dynamic routing

2. **Short Term (Should Do)**
   - [ ] Create ActionButton component
   - [ ] Implement ownership middleware
   - [ ] Test all permission flows end-to-end

3. **Medium Term (Nice to Have)**
   - [ ] Complete contract workflow system
   - [ ] Build reusable UI components
   - [ ] Add comprehensive testing

📈 Progress Timeline

```
Nov 15, 2025:
✅ Fixed sidebar routing issue (removed /modules/ prefix)
✅ Created dynamic dashboard with tRPC statistics
✅ Enhanced dashboard with real-time data display
✅ Added recent activity and contract expiration widgets

Previous Work:
✅ Database schema with RBAC models
✅ Permission system core
✅ Role management system
✅ Basic hooks and utilities
```

🎓 Technical Debt & Considerations

Architecture Decisions Needed

1. Finalize folder structure convention
2. Decide on permission naming convention consistency
3. Choose caching strategy for permissions

Performance Considerations

1. Dashboard queries should be optimized with proper indexes
2. Consider implementing Redis for permission caching
3. Add query result caching for statistics

Security Considerations

1. Verify all API endpoints have permission checks
2. Implement rate limiting
3. Add audit logging for permission changes
4. Test for permission bypass vulnerabilities

📝 Notes for Developers

Working with the RBAC System

1. **Adding a New Permission:**
   - Add to `server/rbac/permissions.ts` (PERMISSION_TREE)
   - Add to `scripts/seed/00-permissions.ts`
   - Run seed script
   - Use in UI with `hasPermission("your.permission")`

2. **Adding a New Menu Item:**
   - Add to `lib/dynamicMenuConfig.ts`
   - Specify required permission
   - Sidebar will automatically filter based on user permissions

3. **Protecting API Endpoints:**
   - Use `hasPermission()` middleware in tRPC procedures
   - Example: `.use(hasPermission(PERMISSION_TREE.contractors.view))`

4. **Protecting UI Elements:**
   - Use `usePermissions()` hook
   - Check with `hasPermission("permission.key")`
   - Conditionally render based on result

🔗 Key Files Reference

Core RBAC

- `server/rbac/permissions.ts` - Permission tree definition
- `hooks/use-permissions.ts` - Client-side permission hook

- `server/api/trpc.ts` - tRPC with permission middleware

Configuration

- `lib/dynamicMenuConfig.ts` - Dynamic menu configuration
- `prisma/schema.prisma` - Database schema with RBAC models

UI Components

- `components/layout/sidebar.tsx` - Dynamic sidebar
- `app/(dashboard)/dashboard/page.tsx` - Dashboard with stats

API Routers

- `server/api/routers/dashboard.ts` - Dashboard statistics
- `server/api/routers/roleManagement.ts` - Role CRUD (if exists)
- `server/api/root.ts` - Router registry

Seeding

- `scripts/seed/00-permissions.ts` - Permission seed
- `scripts/seed/01-roles.ts` - Role seed

📞 Support & Questions

For questions about RBAC implementation, refer to:
1. This progress document (RBAC_PROGRESS.md)
2. The roadmap guide (/home/ubuntu/Uploads/guide.pdf)
3. NextAuth documentation for session management
4. tRPC documentation for API patterns

Last Review Date: November 15, 2025
Reviewed By: AI Development Assistant
Status: Document Created - Initial Progress Assessment Complete