DAY 2: Admin Management Module - Complete Requirements & Checklist

📋 Overview

This document outlines all tasks needed to complete the Admin Management module based on your user stories and the DAY 2 plan.

🎯 User Stories & Requirements

User Story 1: User Profile Display (Customer & Admin)

Status: ✅ BACKEND DONE | ⏳ FRONTEND NEEDED

User Profile Features Required:

- [ ] Display user profile information (name, email, phone, DOB)
- [ ] Edit user profile (email, phone, DOB, name)
- [ ] Change password
- [ ] Profile avatar/image upload
- [ ] Both customer and admin should have access

Backend Endpoints Available ✅:

```
GET  /profile/                     - Get current user profile
PUT  /profile/                     - Update user profile
POST /profile/image                - Upload profile image
POST /profile/change-password      - Change password
```

Angular Components Needed:

- [ ] `ProfileComponent` - Display profile information
- [ ] `EditProfileComponent` - Edit profile modal/form
- [ ] `ChangePasswordComponent` - Change password form

- [ ] `ProfileService` - HTTP calls to backend

User Story 2: Admin-Specific Features

2.1 Admin List View

Description: Main admin should view all admins with filtering and search

Backend Status: ✅ DONE

- Endpoint: `GET /users/list` (Backend has filtering support)

Angular Components Needed:

- [ ] `AdminListComponent` - Table display with:
    - [ ] Columns: ID, Name, Email, Phone, Role, Status, Created Date
    - [ ] Sorting by any column
    - [ ] Filtering by:
        - [ ] Role (super_admin, normal_admin, content_admin, etc.)
        - [ ] Status (active, inactive, suspended)
        - [ ] Date range (created date)
    - [ ] Search functionality (by name, email)
    - [ ] Pagination
    - [ ] Protected route: `ADMIN_CREATION:READ`
    - [ ] Action buttons: View, Edit, Suspend, Delete

Backend Endpoints Required:

```
GET /users/list?role=admin&status=active&page=1&limit=10&search=john
```

2.2 Admin Creation

Description: Main admin creates new admin users with role assignment

Backend Status: ✅ DONE

- Endpoint: `POST /users/create` or similar (check backend)

Angular Components Needed:

- [ ] `CreateAdminComponent` - Form with:
    - [ ] Form fields:
        - [ ] Email (required, unique validation)
        - [ ] Name (required)
        - [ ] Phone Number (optional)
        - [ ] Role Selection (dropdown):
            - [ ] super_admin
            - [ ] normal_admin
            - [ ] content_admin
            - [ ] sales_admin
            - [ ] Support_admin
        - [ ] Status (active/inactive)
        - [ ] Initial Password (generated or user input)
    - [ ] Form validation
    - [ ] Real-time email uniqueness check
    - [ ] Submit button protection: `ADMIN_CREATION:WRITE`
    - [ ] Success/error notifications
    - [ ] Modal or separate page

Backend Endpoint Required:

```
POST /users/create
Body: {
  "email": "admin@example.com",
  "name": "John Doe",
  "phone": "+1234567890",
  "role_id": 2,
  "status": "active"
}
```

2.3 Admin Edit

Description: Main admin edits existing admin details and role

Backend Status: ✅ Likely done (check endpoint)

Angular Components Needed:

- [ ] `EditAdminComponent` - Form to edit:
    - [ ] Name
    - [ ] Phone Number
    - [ ] Role (dropdown)
    - [ ] Status (active/inactive)
    - [ ] Save button protection: `ADMIN_CREATION:WRITE`
    - [ ] Success/error handling
    - [ ] Read-only fields: email, created_at

Backend Endpoint Required:

```
PUT /users/{user_id}/
Body: {
  "name": "Jane Doe",
  "phone": "+1234567890",
  "role_id": 3,
  "status": "active"
}
```

2.4 Admin Suspension

Description: Admin with `admin_creation` permission can suspend other admins

Backend Status: ⏳ VERIFY

- Need endpoint to suspend/deactivate admin

Angular Features Needed:

- [ ] Suspend button in admin list
- [ ] Confirmation modal before suspension
- [ ] Permission check: `ADMIN_CREATION:WRITE`
- [ ] Update UI after suspension

Backend Endpoint Required:

```
PUT /users/{user_id}/suspend
OR
PUT /users/{user_id}/status
Body: { "status": "suspended" }
```

2.5 Create Roles & Assign Permissions

Description: Main admin creates new roles and assigns permissions

Backend Status: ✅ DONE

- Endpoints available in `roles_and_permissions.py`

Angular Components Needed:

- [ ] `RoleManagementComponent` - List all roles
- [ ] `CreateRoleComponent` - Form to create role:
  - [ ] Role name (required)
  - [ ] Description (optional)
  - [ ] Permission selector (multi-select):
    - [ ] Display all available permissions
    - [ ] Checkboxes or toggle switches
    - [ ] Grouped by resource (ADMIN_CREATION, ROOM_MANAGEMENT, etc.)
  - [ ] Save button protection: `ADMIN_CREATION:WRITE`
- [ ] `EditRoleComponent` - Edit role and permissions
- [ ] `RoleService` - HTTP calls to backend

Backend Endpoints Available ✅:

```
GET  /roles                                 - List all roles
POST /roles                                 - Create role
PUT  /roles/{role_id}/permissions          - Assign permissions
GET  /roles/{role_id}/permissions          - Get role permissions
GET  /permissions                           - List all permissions
```

2.6 Update Admin Role

Description: Main admin can change the role of another admin

Backend Status: ✅ DONE

- Handled by admin edit endpoint with role_id update

Angular Features Needed:

- [ ] Role change already covered in EditAdminComponent
- [ ] Permission guard: `ADMIN_CREATION:WRITE`

2.7 Admin Self-Profile Management

Description: Logged-in admin can change their own details and password

Backend Status: ✅ DONE

Angular Components Needed:

- [ ] `AdminProfileComponent` - Same as regular profile but accessed from admin menu
- [ ] Features:
    - [ ] Edit name, phone, DOB, email
    - [ ] Change password
    - [ ] Profile picture upload
    - [ ] Available to all logged-in users (admin/customer)

Backend Endpoints Available ✅:

```
GET  /profile/                    - Get profile
PUT  /profile/                    - Update profile
POST /profile/image               - Upload image
POST /profile/change-password     - Change password
```

🛠️ Technical Implementation Checklist

Backend Verification Tasks

- [ ] Verify all user management endpoints exist
- [ ] Verify suspension endpoint exists
- [ ] Verify permission endpoints work correctly
- [ ] Test all endpoints for permission validation
- [ ] Verify responses match expected format
- [ ] Check error handling for edge cases

Commands to verify:

```
# List endpoints
curl http://localhost:8000/openapi.json | jq '.paths | keys'

# Test endpoints
curl -H "Authorization: Bearer YOUR_TOKEN" http://localhost:8000/users/list
curl -H "Authorization: Bearer YOUR_TOKEN" http://localhost:8000/roles
curl -H "Authorization: Bearer YOUR_TOKEN" http://localhost:8000/permissions
```

Frontend Core Infrastructure Needed

1. Permission Guard Service ✅

- [ ] Verify `PermissionGuard` exists
- [ ] Has `canActivate()` checking for `ADMIN_CREATION:READ/WRITE`
- [ ] Works with `ADMIN_CREATION:*` scope

2. Permission Directive

- [ ] [ ] `*appHasPermission="'ADMIN_CREATION:WRITE'"` directive exists
- [ ] Can hide/show buttons based on permission
- [ ] Can enable/disable buttons

3. Authentication Service

- [ ] Verify stores current user permissions
- [ ] Has method to check single permission: `hasPermission('ADMIN_CREATION:READ')`
- [ ] Has method to check multiple permissions

- [ ] Updates permissions after role change

4. HTTP Interceptor

- [ ] Attaches JWT token to all requests
- [ ] Handles token refresh
- [ ] Handles 401/403 errors

Angular Component Structure Required

```
src/app/features/
├── admin/
│   ├── admin-routing.module.ts
│   ├── admin.module.ts
│   ├── admin-list/
│   │   ├── admin-list.component.ts
│   │   ├── admin-list.component.html
│   │   ├── admin-list.component.css
│   │   └── admin-list.component.spec.ts
│   ├── admin-create/
│   │   ├── admin-create.component.ts
│   │   ├── admin-create.component.html
│   │   └── admin-create.component.css
│   ├── admin-edit/
│   │   ├── admin-edit.component.ts
│   │   ├── admin-edit.component.html
│   │   └── admin-edit.component.css
│   ├── role-management/
│   │   ├── role-list.component.ts
│   │   ├── role-create.component.ts
│   │   ├── role-edit.component.ts
│   │   └── role-management.service.ts
│   └── admin.service.ts
├── profile/
│   ├── profile/
│   │   ├── profile.component.ts
│   │   ├── profile.component.html
│   │   └── profile.component.css
│   ├── edit-profile/
│   │   ├── edit-profile.component.ts
│   │   ├── edit-profile.component.html
│   │   └── edit-profile.component.css
│   ├── change-password/
│   │   ├── change-password.component.ts
│   │   ├── change-password.component.html
│   │   └── change-password.component.css
│   └── profile.service.ts
```

Services Required

1. **AdminService** (New)

```
// Methods needed:
- getAllAdmins(filters?: AdminFilters): Observable<AdminListResponse>
- getAdminById(id: number): Observable<Admin>
- createAdmin(data: CreateAdminRequest): Observable<Admin>
- updateAdmin(id: number, data: UpdateAdminRequest): Observable<Admin>
- suspendAdmin(id: number): Observable<any>
- deleteAdmin(id: number): Observable<any>
```

2. **RoleService** (New)

```
// Methods needed:
- getAllRoles(): Observable<Role[]>
- getRoleById(id: number): Observable<Role>
- createRole(data: CreateRoleRequest): Observable<Role>
- updateRole(id: number, data: UpdateRoleRequest): Observable<Role>
- deleteRole(id: number): Observable<any>
- assignPermissionsToRole(roleId: number, permissionIds: number[]): Observable<any>
- getRolePermissions(roleId: number): Observable<Permission[]>
```

3. **ProfileService** (New)

```
// Methods needed:
- getProfile(): Observable<Profile>
- updateProfile(data: UpdateProfileRequest): Observable<Profile>
- uploadProfileImage(file: File): Observable<Profile>
- changePassword(data: ChangePasswordRequest): Observable<any>
```

4. **PermissionService** (Verify/Update)

```
// Methods needed:
- getAllPermissions(): Observable<Permission[]>
- hasPermission(permission: string): boolean
- hasAnyPermission(permissions: string[]): boolean
- hasAllPermissions(permissions: string[]): boolean
- getPermissionsByResource(resource: string): Observable<Permission[]>
```

📅 Implementation Order (Recommended)

Phase 1: Infrastructure (Day 1-1.5)

- [ ] Verify backend endpoints
- [ ] Verify/create PermissionGuard
- [ ] Verify/create Permission Directive
- [ ] Create/update PermissionService

Phase 2: Services (Day 1.5-2)

- [ ] Create AdminService
- [ ] Create RoleService
- [ ] Create ProfileService

Phase 3: Profile Module (Day 2-2.5)

- [ ] Create ProfileComponent
- [ ] Create EditProfileComponent (modal/form)
- [ ] Create ChangePasswordComponent (modal/form)
- [ ] Add routes for profile

Phase 4: Admin Management (Day 2.5-4)

- [ ] Create AdminListComponent (table, filter, search)
- [ ] Create CreateAdminComponent (form)
- [ ] Create EditAdminComponent (form)
- [ ] Add routes for admin management

Phase 5: Role Management (Day 4-5)

- [ ] Create RoleListComponent
- [ ] Create CreateRoleComponent (with permission multi-select)
- [ ] Create EditRoleComponent
- [ ] Add routes for role management

🔒 Permission Scopes to Check

```
ADMIN_CREATION:READ     - View admin list and details
ADMIN_CREATION:WRITE    - Create, edit, suspend admins
ADMIN_CREATION:DELETE   - Delete admins (if applicable)
ROLE_MANAGEMENT:READ    - View roles
ROLE_MANAGEMENT:WRITE   - Create/edit roles
```

🧪 Testing Checklist

Unit Tests

- [ ] AdminService: CRUD operations
- [ ] RoleService: CRUD operations
- [ ] ProfileService: CRUD operations
- [ ] Permission checks in all components

E2E Tests

- [ ] Admin can view list of admins
- [ ] Admin can create new admin
- [ ] Admin can edit admin
- [ ] Admin can suspend admin
- [ ] Admin can manage roles
- [ ] Admin can assign permissions
- [ ] User can view and edit their profile
- [ ] User can change password
- [ ] Permission guards work correctly

❓ Questions to Verify with Backend Team

1. Is there a suspension endpoint, or should status be updated?
2. What's the exact format for filtering users (query params)?
3. Are there role IDs we should know about?

4. What permissions are available in the system?
5. What's the initial password policy for new admins?
6. Can admins update their own role?
7. Are there audit logs for admin actions?

📌 Notes

- All user-facing features require corresponding Angular components
- Every action modifying data needs permission checks
- UI should reflect user permissions (hide buttons for unauthorized users)
- All forms need validation and error handling
- Implement loading states for async operations
- Add success/error toast notifications
- Consider pagination for large lists
- Implement proper error handling for all HTTP calls