# Module Management API Documentation

Complete API reference for the Module Management System that provides dynamic control over frontend navigation, permissions, and user experience customization.

## 🎯 Base URL

/api/v1/modules

## 🔐 Authentication

All endpoints require Bearer token authentication:

Authorization: Bearer <your-jwt-token>

## 📋 Module Management Endpoints

### Create Module

POST /api/v1/modules

**Description:** Create a new application module

**Required Permission:** modules:admin

**Request Body:**

{  
 "name": "patch\_management",  
 "description": "Patch deployment and management system",  
 "enabled": false  
}

**Response:**

{  
 "message": "Module created successfully",  
 "data": {  
 "id": 1,  
 "name": "patch\_management",  
 "description": "Patch deployment and management system",  
 "enabled": false,  
 "createdAt": "2024-01-15T10:30:00Z",  
 "updatedAt": "2024-01-15T10:30:00Z"  
 }  
}

### Get All Modules

GET /api/v1/modules?enabled=true&search=patch&page=1&limit=20&sortBy=name&sortOrder=asc

**Description:** Retrieve all modules with filtering and pagination

**Required Permission:** modules:read

**Query Parameters:** - enabled (boolean, optional): Filter by enabled status - search (string, optional): Search in name and description - page (integer, optional, default: 1): Page number - limit (integer, optional, default: 20, max: 100): Results per page - sortBy (string, optional, default: name): Sort field (name, enabled, createdAt, updatedAt) - sortOrder (string, optional, default: asc): Sort order (asc, desc)

**Response:**

{  
 "message": "Modules retrieved successfully",  
 "data": [  
 {  
 "id": 1,  
 "name": "patch\_management",  
 "description": "Patch deployment and management system",  
 "enabled": true,  
 "createdAt": "2024-01-15T10:30:00Z",  
 "updatedAt": "2024-01-15T10:30:00Z"  
 }  
 ],  
 "pagination": {  
 "page": 1,  
 "limit": 20,  
 "totalCount": 1,  
 "totalPages": 1,  
 "hasNextPage": false,  
 "hasPreviousPage": false  
 }  
}

### Get Module by ID

GET /api/v1/modules/{moduleId}

**Description:** Get detailed module information including navigation and permissions

**Required Permission:** modules:read

**Response:**

{  
 "message": "Module retrieved successfully",  
 "data": {  
 "id": 1,  
 "name": "patch\_management",  
 "description": "Patch deployment and management system",  
 "enabled": true,  
 "navigation": [  
 {  
 "id": 1,  
 "navLabel": "Patch Management",  
 "navPath": "/patches",  
 "navIcon": "update",  
 "navOrder": 1,  
 "parentId": null,  
 "isVisible": true,  
 "requiresPermission": null  
 }  
 ],  
 "permissions": [  
 {  
 "roleId": 1,  
 "roleName": "admin",  
 "canView": true,  
 "canCreate": true,  
 "canEdit": true,  
 "canDelete": true,  
 "canAdmin": true  
 }  
 ],  
 "dependencies": [],  
 "settings": []  
 }  
}

### Update Module

PUT /api/v1/modules/{moduleId}

**Description:** Update module information

**Required Permission:** modules:admin

**Request Body:**

{  
 "name": "patch\_management\_v2",  
 "description": "Enhanced patch deployment and management system",  
 "enabled": true  
}

### Delete Module

DELETE /api/v1/modules/{moduleId}

**Description:** Delete a module (only if no dependencies exist)

**Required Permission:** modules:admin

**Response:**

{  
 "message": "Module deleted successfully",  
 "data": {  
 "success": true,  
 "message": "Module deleted successfully"  
 }  
}

### Toggle Module Status

POST /api/v1/modules/{moduleId}/toggle

**Description:** Enable or disable a module

**Required Permission:** modules:admin

**Response:**

{  
 "message": "Module enabled successfully",  
 "data": {  
 "id": 1,  
 "name": "patch\_management",  
 "enabled": true,  
 "updatedAt": "2024-01-15T11:00:00Z"  
 }  
}

## 🧭 Navigation Management Endpoints

### Create Navigation Item

POST /api/v1/modules/navigation

**Description:** Create a new navigation item for a module

**Required Permission:** modules:admin

**Request Body:**

{  
 "moduleId": 1,  
 "navLabel": "Patch Dashboard",  
 "navPath": "/patches/dashboard",  
 "navIcon": "dashboard",  
 "navOrder": 1,  
 "parentId": null,  
 "isVisible": true,  
 "requiresPermission": "patches:read"  
}

### Get Module Navigation

GET /api/v1/modules/{moduleId}/navigation

**Description:** Get hierarchical navigation structure for a module

**Required Permission:** modules:read

**Response:**

{  
 "message": "Module navigation retrieved successfully",  
 "data": [  
 {  
 "id": 1,  
 "navLabel": "Patch Management",  
 "navPath": "/patches",  
 "navIcon": "update",  
 "navOrder": 1,  
 "children": [  
 {  
 "id": 2,  
 "navLabel": "Patch Dashboard",  
 "navPath": "/patches/dashboard",  
 "navIcon": "dashboard",  
 "navOrder": 1,  
 "children": []  
 }  
 ]  
 }  
 ]  
}

### Get User Navigation

GET /api/v1/modules/navigation/user

**Description:** Get personalized navigation for the current user based on role permissions and preferences

**Authentication:** Required (uses current user context)

**Response:**

{  
 "message": "User navigation retrieved successfully",  
 "data": [  
 {  
 "moduleId": 1,  
 "moduleName": "patch\_management",  
 "navId": 1,  
 "navLabel": "Patch Management",  
 "navPath": "/patches",  
 "navIcon": "update",  
 "navOrder": 1,  
 "canView": true,  
 "canCreate": true,  
 "canEdit": true,  
 "canDelete": false,  
 "canAdmin": false,  
 "children": []  
 }  
 ]  
}

## 🔐 Permission Management Endpoints

### Set Role Module Permissions

PUT /api/v1/modules/roles/{roleId}/modules/{moduleId}/permissions

**Description:** Set permissions for a specific role on a specific module

**Required Permission:** modules:admin

**Request Body:**

{  
 "canView": true,  
 "canCreate": true,  
 "canEdit": true,  
 "canDelete": false,  
 "canAdmin": false  
}

**Response:**

{  
 "message": "Role module permissions set successfully",  
 "data": {  
 "id": 1,  
 "roleId": 2,  
 "moduleId": 1,  
 "canView": true,  
 "canCreate": true,  
 "canEdit": true,  
 "canDelete": false,  
 "canAdmin": false,  
 "createdAt": "2024-01-15T10:30:00Z",  
 "updatedAt": "2024-01-15T11:00:00Z"  
 }  
}

### Get Role Module Permissions

GET /api/v1/modules/roles/{roleId}/modules/{moduleId}/permissions

**Description:** Get permissions for a specific role on a specific module

**Required Permission:** modules:read

**Response:**

{  
 "message": "Role module permissions retrieved successfully",  
 "data": {  
 "canView": true,  
 "canCreate": true,  
 "canEdit": true,  
 "canDelete": false,  
 "canAdmin": false  
 }  
}

### Check User Module Permission

GET /api/v1/modules/{moduleId}/permissions/{permission}/check

**Description:** Check if the current user has a specific permission for a module

**Parameters:** - moduleId (integer): Module ID - permission (string): Permission to check (canView, canCreate, canEdit, canDelete, canAdmin)

**Response:**

{  
 "message": "User module permission checked successfully",  
 "data": {  
 "userId": 123,  
 "moduleId": 1,  
 "permission": "canEdit",  
 "hasPermission": true  
 }  
}

## 📊 Usage Examples

### Frontend Integration Example

// React component for dynamic navigation  
import React, { useState, useEffect } from 'react';  
  
const DynamicNavigation = () => {  
 const [navigation, setNavigation] = useState([]);  
 const [loading, setLoading] = useState(true);  
  
 useEffect(() => {  
 fetchUserNavigation();  
 }, []);  
  
 const fetchUserNavigation = async () => {  
 try {  
 const response = await fetch('/api/v1/modules/navigation/user', {  
 headers: {  
 'Authorization': `Bearer ${localStorage.getItem('token')}`  
 }  
 });  
 const data = await response.json();  
 setNavigation(data.data);  
 } catch (error) {  
 console.error('Error fetching navigation:', error);  
 } finally {  
 setLoading(false);  
 }  
 };  
  
 const renderNavigationItem = (item) => (  
 <li key={item.navId}>  
 <a href={item.navPath} className="nav-link">  
 <i className={`icon-${item.navIcon}`}></i>  
 {item.navLabel}  
 </a>  
 {item.children && item.children.length > 0 && (  
 <ul className="nav-submenu">  
 {item.children.map(renderNavigationItem)}  
 </ul>  
 )}  
 </li>  
 );  
  
 if (loading) return <div>Loading navigation...</div>;  
  
 return (  
 <nav className="sidebar-nav">  
 <ul className="nav">  
 {navigation.map(renderNavigationItem)}  
 </ul>  
 </nav>  
 );  
};  
  
export default DynamicNavigation;

### Permission-Based UI Component

// React hook for permission checking  
import { useState, useEffect } from 'react';  
  
const useModulePermission = (moduleId, permission) => {  
 const [hasPermission, setHasPermission] = useState(false);  
 const [loading, setLoading] = useState(true);  
  
 useEffect(() => {  
 checkPermission();  
 }, [moduleId, permission]);  
  
 const checkPermission = async () => {  
 try {  
 const response = await fetch(  
 `/api/v1/modules/${moduleId}/permissions/${permission}/check`,  
 {  
 headers: {  
 'Authorization': `Bearer ${localStorage.getItem('token')}`  
 }  
 }  
 );  
 const data = await response.json();  
 setHasPermission(data.data.hasPermission);  
 } catch (error) {  
 console.error('Error checking permission:', error);  
 setHasPermission(false);  
 } finally {  
 setLoading(false);  
 }  
 };  
  
 return { hasPermission, loading };  
};  
  
// Usage in component  
const PatchManagementActions = () => {  
 const { hasPermission: canCreate } = useModulePermission(1, 'canCreate');  
 const { hasPermission: canDelete } = useModulePermission(1, 'canDelete');  
  
 return (  
 <div className="action-buttons">  
 {canCreate && (  
 <button className="btn btn-primary">Create Patch</button>  
 )}  
 {canDelete && (  
 <button className="btn btn-danger">Delete Patch</button>  
 )}  
 </div>  
 );  
};

### Administrative Module Management

// Admin component for module management  
const ModuleManagement = () => {  
 const [modules, setModules] = useState([]);  
  
 const toggleModuleStatus = async (moduleId) => {  
 try {  
 const response = await fetch(`/api/v1/modules/${moduleId}/toggle`, {  
 method: 'POST',  
 headers: {  
 'Authorization': `Bearer ${localStorage.getItem('token')}`  
 }  
 });  
   
 if (response.ok) {  
 const data = await response.json();  
 // Update local state  
 setModules(prev => prev.map(module =>   
 module.id === moduleId   
 ? { ...module, enabled: data.data.enabled }  
 : module  
 ));  
 }  
 } catch (error) {  
 console.error('Error toggling module:', error);  
 }  
 };  
  
 const setRolePermissions = async (roleId, moduleId, permissions) => {  
 try {  
 const response = await fetch(  
 `/api/v1/modules/roles/${roleId}/modules/${moduleId}/permissions`,  
 {  
 method: 'PUT',  
 headers: {  
 'Content-Type': 'application/json',  
 'Authorization': `Bearer ${localStorage.getItem('token')}`  
 },  
 body: JSON.stringify(permissions)  
 }  
 );  
   
 if (response.ok) {  
 console.log('Permissions updated successfully');  
 }  
 } catch (error) {  
 console.error('Error updating permissions:', error);  
 }  
 };  
  
 return (  
 <div className="module-management">  
 {modules.map(module => (  
 <div key={module.id} className="module-card">  
 <h3>{module.name}</h3>  
 <p>{module.description}</p>  
 <button   
 onClick={() => toggleModuleStatus(module.id)}  
 className={`btn ${module.enabled ? 'btn-success' : 'btn-secondary'}`}  
 >  
 {module.enabled ? 'Enabled' : 'Disabled'}  
 </button>  
 </div>  
 ))}  
 </div>  
 );  
};

## 🚀 Getting Started

1. **Run the migration script:**

* node scripts/migrate\_module\_tables.js

1. **Test the API endpoints:**

* # Get user navigation  
  curl -H "Authorization: Bearer YOUR\_TOKEN" \  
   http://localhost:3000/api/v1/modules/navigation/user  
    
  # Check permission  
  curl -H "Authorization: Bearer YOUR\_TOKEN" \  
   http://localhost:3000/api/v1/modules/1/permissions/canView/check

1. **Integrate with your frontend:**
   * Use the user navigation endpoint to build dynamic menus
   * Implement permission checking for UI elements
   * Set up module-based routing and access control

The Module Management System provides a robust foundation for dynamic, role-based application control with comprehensive customization and monitoring capabilities.