# Template Starter Kit for SAAS platform and Digital Application

Design a plat for a template starter-kit platform for SAAS and Digital Application.

The tech stack:

1. Backend - NestJS
2. Database - PostreSQL
3. Admin Dashboard (Admin Frontend) - NextJS
4. Design System and React Components - [https://ui.shadcn.com/](https://ui.shadcn.com/blocks)
5. Project Code Management - Mono Repo
6. Infrastructure - IaaC - Terraform (in a folder in the root of the project)
7. Graphic resources for use in the project (logo, images, other) - resources folder in the root of the project
8. AI Code generation - claude-code
9. AI task management - [TODO.md](http://todo.md) inside a tasks folder. Create a workplan and update the workplan and task status while developing and building the projects.
10. Main code instructions file - [CLAUDE.md](http://claude.md) in the project folder root
11. Style instructions - in the folder Style - STYLE\_GUIDE.md
12. Create Component Storybook - for consistency reference
13. Maintain UI\_CHANGE\_LOG.md in the style folder
14. Development environments - dev, stage, prod. Use .env files
15. ORM - use typeORM
16. Frontend state management - use Redux
17. Front end networking - use Axios
18. For local development use docker-comose
19. Code management - git
20. Commit changes and create develop branch
21. Merge to main branch when done
22. Push to remote branch, work with GitHub - Ask the developer for remote branch information when setting or starting the development of the project.
23. Authorization - Use ABAC
24. Project Apps (Backend, Admin Dashboard, frontend) are in the Apps folder.
25. Coding language - Typescript
26. Code documentation - us JSDOC, . Document every function, class, module, enum, also add inline documentation for complex or important logic
27. API Documentation and access - SWAGGER
28. Use AWS Cognito as the Authentication platform. Use user-pool named “next-dev”

## UX UI Guide for Consistent SaaS Dashboard

Create the NextJS project, with tailwind, typescript

Install the shdcn/ui

Configure the shadcn/ui

// components.json

{

"style": "default",

"rsc": true,

"tsx": true,

"tailwind": {

"config": "tailwind.config.ts",

"css": "app/globals.css",

"baseColor": "slate",

"cssVariables": true

},

"aliases": {

"components": "@/components",

"utils": "@/lib/utils"

}

}

## Create COMPONENETS\_REFERENCE.md in the style folder

# COMPONENTS\_REFERENCE.md

## Available shadcn/ui Components

- Button: Primary, Secondary, Destructive, Outline, Ghost, Link

- Card: Card, CardHeader, CardTitle, CardDescription, CardContent, CardFooter

- Input: Standard text input with label

- Select: Dropdown selection

- Table: Data display with sorting

- Dialog: Modal windows

- Toast: Notifications

## Import Pattern

```typescript

import { Button } from "@/components/ui/button"

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card"

## Components installation

# Install components as needed

npx shadcn-ui@latest add button

npx shadcn-ui@latest add card

npx shadcn-ui@latest add table

npx shadcn-ui@latest add form

npx shadcn-ui@latest add dashboard

## 

## Template Prompts and instructions

### Dashboard Page Template

"Create a dashboard page using shadcn/ui components. Include:

* Page layout with sidebar navigation
* Card components for metrics
* Table for recent data
* Use only installed shadcn components from COMPONENTS\_REFERENCE.md"

### Form Template

"Build a form using shadcn/ui Form component with:

* Proper validation using zod
* Toast notifications for success/error
* Consistent spacing using space-y-4"

## Style Guide

// styles/constants.ts

export const SPACING = {

page: "p-6",

card: "p-4",

section: "space-y-6",

form: "space-y-4"

}

export const LAYOUTS = {

dashboard: "grid gap-4 md:grid-cols-2 lg:grid-cols-4",

form: "max-w-2xl mx-auto",

table: "w-full"

}

## Layout Patterns

// layouts/DashboardLayout.tsx

export default function DashboardLayout({ children }) {

return (

<div className="flex h-screen bg-gray-50">

<Sidebar />

<main className="flex-1 overflow-y-auto p-6">

{children}

</main>

</div>

)

}

## Component Patterns

// components/MetricCard.tsx

export function MetricCard({ title, value, change }) {

return (

<Card>

<CardHeader className="pb-2">

<CardTitle className="text-sm font-medium text-gray-600">

{title}

</CardTitle>

</CardHeader>

<CardContent>

<div className="text-2xl font-bold">{value}</div>

<p className="text-xs text-gray-500">{change}</p>

</CardContent>

</Card>

)

}

## Use Consistent Spacing

const spacing = {

section: "mb-8",

component: "mb-4",

element: "mb-2"

}

## Create Base Components Structure

// templates/CardTemplate.tsx

export const CardTemplate = ({ title, children }) => (

<Card className="p-6">

<h3 className="text-lg font-semibold mb-4">{title}</h3>

{children}

</Card>

)

Put base components in a “base” folder in the components folder.

## Use CSS Variables for Theming

/\* globals.css \*/

:root {

--primary: 243 80% 62%;

--secondary: 263 70% 65%;

--background: 0 0% 100%;

--foreground: 224 71% 4%;

}

## MAIN Layouts Reference:

Dashboard layout

<https://ui.shadcn.com/blocks>

Use dashboard-01 option

Sidebar

Use sidebar-07

Authentication

<https://ui.shadcn.com/blocks/authentication>

Use login-04

Other components

<https://ui.shadcn.com/docs/components>

Shadcn Theme

Use Default theme.

https://ui.shadcn.com/themes

### Style Guide

Use this style guide,

Compatible with Tailwind v4

app/global.css

:root {

--radius: 0.65rem;

--background: oklch(1 0 0);

--foreground: oklch(0.145 0 0);

--card: oklch(1 0 0);

--card-foreground: oklch(0.145 0 0);

--popover: oklch(1 0 0);

--popover-foreground: oklch(0.145 0 0);

--primary: oklch(0.205 0 0);

--primary-foreground: oklch(0.985 0 0);

--secondary: oklch(0.97 0 0);

--secondary-foreground: oklch(0.205 0 0);

--muted: oklch(0.97 0 0);

--muted-foreground: oklch(0.556 0 0);

--accent: oklch(0.97 0 0);

--accent-foreground: oklch(0.205 0 0);

--destructive: oklch(0.577 0.245 27.325);

--border: oklch(0.922 0 0);

--input: oklch(0.922 0 0);

--ring: oklch(0.708 0 0);

--chart-1: oklch(0.646 0.222 41.116);

--chart-2: oklch(0.6 0.118 184.704);

--chart-3: oklch(0.398 0.07 227.392);

--chart-4: oklch(0.828 0.189 84.429);

--chart-5: oklch(0.769 0.188 70.08);

--radius: 0.625rem;

--sidebar: oklch(0.985 0 0);

--sidebar-foreground: oklch(0.145 0 0);

--sidebar-primary: oklch(0.205 0 0);

--sidebar-primary-foreground: oklch(0.985 0 0);

--sidebar-accent: oklch(0.97 0 0);

--sidebar-accent-foreground: oklch(0.205 0 0);

--sidebar-border: oklch(0.922 0 0);

--sidebar-ring: oklch(0.708 0 0);

}

.dark {

--background: oklch(0.145 0 0);

--foreground: oklch(0.985 0 0);

--card: oklch(0.205 0 0);

--card-foreground: oklch(0.985 0 0);

--popover: oklch(0.205 0 0);

--popover-foreground: oklch(0.985 0 0);

--primary: oklch(0.922 0 0);

--primary-foreground: oklch(0.205 0 0);

--secondary: oklch(0.269 0 0);

--secondary-foreground: oklch(0.985 0 0);

--muted: oklch(0.269 0 0);

--muted-foreground: oklch(0.708 0 0);

--accent: oklch(0.269 0 0);

--accent-foreground: oklch(0.985 0 0);

--destructive: oklch(0.704 0.191 22.216);

--border: oklch(1 0 0 / 10%);

--input: oklch(1 0 0 / 15%);

--ring: oklch(0.556 0 0);

--chart-1: oklch(0.488 0.243 264.376);

--chart-2: oklch(0.696 0.17 162.48);

--chart-3: oklch(0.769 0.188 70.08);

--chart-4: oklch(0.627 0.265 303.9);

--chart-5: oklch(0.645 0.246 16.439);

--sidebar: oklch(0.205 0 0);

--sidebar-foreground: oklch(0.985 0 0);

--sidebar-primary: oklch(0.488 0.243 264.376);

--sidebar-primary-foreground: oklch(0.985 0 0);

--sidebar-accent: oklch(0.269 0 0);

--sidebar-accent-foreground: oklch(0.985 0 0);

--sidebar-border: oklch(1 0 0 / 10%);

--sidebar-ring: oklch(0.556 0 0);

}

# Multi-Organization ABAC System Design & Implementation Plan

The document outlines the design and implementation of a dynamic, multi-organization Attribute-Based Access Control (ABAC) system built on NestJS and PostgreSQL. The system provides granular authorization capabilities across hierarchical organizations with cross-organizational access controls and a comprehensive admin dashboard for policy management.

## 1. System Architecture Overview

### 1.1 High-Level Architecture

┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐

│ Admin UI │ │ Client Apps │ │ Mobile Apps │

│ (React NextJS) │ │ (Various) │ │ (Various) │

└─────────────────┘ └─────────────────┘ └─────────────────┘

│ │ │

└───────────────────────┼───────────────────────┘

│

┌─────────────────┐

│ API Gateway │

│ (NestJS) │

└─────────────────┘

│

┌───────────────────────┼───────────────────────┐

│ │ │

┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐

│ Auth Service │ │ ABAC Engine │ │ Resource Service│

│ (NestJS) │ │ (NestJS) │ │ (NestJS) │

└─────────────────┘ └─────────────────┘ └─────────────────┘

│ │ │

└───────────────────────┼───────────────────────┘

│

┌─────────────────┐

│ PostgreSQL │

│ Database │

└─────────────────┘

### 1.2 Core Components

1. **API Gateway**: Central entry point with organization routing
2. **Authentication Service**: JWT-based auth with organization context
3. **ABAC Engine**: Policy evaluation and decision making
4. **Resource Service**: Protected resource management
5. **Admin Dashboard (React NextJS)**: Policy and attribute management UI
6. **Policy Decision Point (PDP)**: Core authorization logic
7. **Policy Administration Point (PAP)**: Policy management interface

## 2. Hierarchical Organization Model

### 2.1 Organization Structure

Organizations support nested hierarchical structures where any organization can contain child organizations (departments, divisions, teams, etc.).

Root Organization (Company)

├── Sales Department

│ ├── North America Sales

│ │ ├── US East Coast

│ │ └── US West Coast

│ └── International Sales

│ ├── Europe Division

│ └── Asia Pacific Division

├── Engineering Department

│ ├── Backend Team

│ ├── Frontend Team

│ └── DevOps Team

└── Finance Department

├── Accounting

└── Budgeting

### 2.2 Enhanced Database Design

-- Organizations with hierarchical support

CREATE TABLE organizations (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

name VARCHAR(255) NOT NULL,

slug VARCHAR(100) NOT NULL,

parent\_organization\_id UUID REFERENCES organizations(id),

organization\_type VARCHAR(50) NOT NULL, -- 'company', 'department', 'team', 'division'

level INTEGER DEFAULT 0, -- Hierarchy level (0 = root)

path TEXT, -- Materialized path for efficient queries (e.g., '/1/2/3/')

settings JSONB DEFAULT '{}',

is\_active BOOLEAN DEFAULT TRUE,

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW(),

UNIQUE(parent\_organization\_id, slug)

);

-- Users with enhanced organization context

CREATE TABLE users (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

organization\_id UUID REFERENCES organizations(id),

primary\_organization\_id UUID REFERENCES organizations(id), -- Main org membership

email VARCHAR(255) UNIQUE NOT NULL,

username VARCHAR(100),

first\_name VARCHAR(100),

last\_name VARCHAR(100),

employee\_id VARCHAR(50),

job\_title VARCHAR(100),

department VARCHAR(100),

manager\_id UUID REFERENCES users(id),

hire\_date DATE,

is\_active BOOLEAN DEFAULT TRUE,

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW()

);

-- User organization memberships (many-to-many)

CREATE TABLE user\_organization\_memberships (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID REFERENCES users(id),

organization\_id UUID REFERENCES organizations(id),

role VARCHAR(100), -- 'member', 'admin', 'manager', 'viewer'

permissions JSONB DEFAULT '{}',

is\_primary BOOLEAN DEFAULT FALSE,

joined\_at TIMESTAMP DEFAULT NOW(),

UNIQUE(user\_id, organization\_id)

);

-- Enhanced attribute definitions with inheritance

CREATE TABLE attribute\_definitions (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

organization\_id UUID REFERENCES organizations(id),

name VARCHAR(100) NOT NULL,

data\_type VARCHAR(50) NOT NULL,

description TEXT,

is\_global BOOLEAN DEFAULT FALSE,

is\_inherited BOOLEAN DEFAULT FALSE, -- Inherited by child organizations

validation\_rules JSONB,

default\_value JSONB,

created\_at TIMESTAMP DEFAULT NOW(),

UNIQUE(organization\_id, name)

);

-- Products (Business Object Example)

CREATE TABLE products (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

organization\_id UUID REFERENCES organizations(id),

name VARCHAR(255) NOT NULL,

description TEXT,

sku VARCHAR(100),

price DECIMAL(10,2),

category VARCHAR(100),

status VARCHAR(50) DEFAULT 'active', -- 'active', 'inactive', 'discontinued'

attributes JSONB DEFAULT '{}',

created\_by UUID REFERENCES users(id),

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW()

);

-- Customers (Business Object Example)

CREATE TABLE customers (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

organization\_id UUID REFERENCES organizations(id),

first\_name VARCHAR(100),

last\_name VARCHAR(100),

company\_name VARCHAR(255),

email VARCHAR(255),

phone VARCHAR(50),

address JSONB,

customer\_type VARCHAR(50) DEFAULT 'individual', -- 'individual', 'business'

status VARCHAR(50) DEFAULT 'active',

credit\_limit DECIMAL(10,2),

attributes JSONB DEFAULT '{}',

created\_by UUID REFERENCES users(id),

assigned\_to UUID REFERENCES users(id), -- Sales rep

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW()

);

-- Orders (Business Object Example)

CREATE TABLE orders (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

organization\_id UUID REFERENCES organizations(id),

order\_number VARCHAR(100) UNIQUE NOT NULL,

customer\_id UUID REFERENCES customers(id),

status VARCHAR(50) DEFAULT 'pending', -- 'pending', 'confirmed', 'shipped', 'delivered', 'cancelled'

total\_amount DECIMAL(10,2),

currency VARCHAR(3) DEFAULT 'USD',

shipping\_address JSONB,

billing\_address JSONB,

attributes JSONB DEFAULT '{}',

created\_by UUID REFERENCES users(id),

assigned\_to UUID REFERENCES users(id), -- Order handler

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW()

);

-- Order Items

CREATE TABLE order\_items (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

order\_id UUID REFERENCES orders(id),

product\_id UUID REFERENCES products(id),

quantity INTEGER NOT NULL,

unit\_price DECIMAL(10,2),

total\_price DECIMAL(10,2),

attributes JSONB DEFAULT '{}'

);

-- Transactions (Business Object Example)

CREATE TABLE transactions (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

organization\_id UUID REFERENCES organizations(id),

transaction\_number VARCHAR(100) UNIQUE NOT NULL,

order\_id UUID REFERENCES orders(id),

customer\_id UUID REFERENCES customers(id),

type VARCHAR(50) NOT NULL, -- 'payment', 'refund', 'adjustment'

amount DECIMAL(10,2) NOT NULL,

currency VARCHAR(3) DEFAULT 'USD',

payment\_method VARCHAR(50), -- 'credit\_card', 'bank\_transfer', 'cash'

status VARCHAR(50) DEFAULT 'pending', -- 'pending', 'completed', 'failed', 'cancelled'

reference\_number VARCHAR(255),

attributes JSONB DEFAULT '{}',

processed\_by UUID REFERENCES users(id),

processed\_at TIMESTAMP,

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW()

);

-- Enhanced policies with hierarchical support

CREATE TABLE policies (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

policy\_set\_id UUID REFERENCES policy\_sets(id),

organization\_id UUID REFERENCES organizations(id),

name VARCHAR(255) NOT NULL,

description TEXT,

effect VARCHAR(10) NOT NULL CHECK (effect IN ('permit', 'deny')),

target JSONB, -- Conditions for when this policy applies

conditions JSONB, -- Additional conditions

inheritance\_mode VARCHAR(20) DEFAULT 'none', -- 'none', 'inherit\_down', 'inherit\_up', 'both'

applies\_to\_children BOOLEAN DEFAULT FALSE,

is\_active BOOLEAN DEFAULT TRUE,

priority INTEGER DEFAULT 0,

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW()

);

-- Organization hierarchy materialized view for performance

CREATE MATERIALIZED VIEW organization\_hierarchy AS

WITH RECURSIVE org\_tree AS (

-- Base case: root organizations

SELECT

id,

name,

slug,

parent\_organization\_id,

organization\_type,

level,

path,

ARRAY[id] as ancestors,

0 as depth

FROM organizations

WHERE parent\_organization\_id IS NULL

UNION ALL

-- Recursive case: child organizations

SELECT

o.id,

o.name,

o.slug,

o.parent\_organization\_id,

o.organization\_type,

o.level,

o.path,

ot.ancestors || o.id,

ot.depth + 1

FROM organizations o

JOIN org\_tree ot ON o.parent\_organization\_id = ot.id

)

SELECT \* FROM org\_tree;

-- Indexes for performance

CREATE INDEX idx\_organizations\_parent ON organizations(parent\_organization\_id);

CREATE INDEX idx\_organizations\_path ON organizations USING GIST(path);

CREATE INDEX idx\_organizations\_type ON organizations(organization\_type);

CREATE INDEX idx\_users\_organization\_id ON users(organization\_id);

CREATE INDEX idx\_users\_primary\_organization ON users(primary\_organization\_id);

CREATE INDEX idx\_user\_org\_memberships\_user ON user\_organization\_memberships(user\_id);

CREATE INDEX idx\_user\_org\_memberships\_org ON user\_organization\_memberships(organization\_id);

CREATE INDEX idx\_products\_organization ON products(organization\_id);

CREATE INDEX idx\_customers\_organization ON customers(organization\_id);

CREATE INDEX idx\_orders\_organization ON orders(organization\_id);

CREATE INDEX idx\_transactions\_organization ON transactions(organization\_id);

CREATE INDEX idx\_policies\_organization ON policies(organization\_id);

CREATE INDEX idx\_policies\_inheritance ON policies(applies\_to\_children) WHERE applies\_to\_children = TRUE;

## 3. Enhanced ABAC Implementation

### 3.1 Hierarchical Policy Evaluation

// Enhanced ABAC service with hierarchical support

@Injectable()

export class HierarchicalAbacService {

constructor(

@InjectRepository(Policy) private policyRepo: Repository<Policy>,

@InjectRepository(Organization) private orgRepo: Repository<Organization>,

private readonly organizationService: OrganizationService,

private readonly policyEvaluator: PolicyEvaluatorService,

) {}

async authorize(request: AuthorizationRequest): Promise<AuthorizationResponse> {

const context = await this.buildHierarchicalContext(request);

const applicablePolicies = await this.findHierarchicalPolicies(context);

return this.evaluatePolicies(applicablePolicies, context);

}

private async buildHierarchicalContext(request: AuthorizationRequest): Promise<AuthorizationContext> {

const user = await this.userService.findById(request.userId);

const userOrganizations = await this.getUserOrganizations(request.userId);

const organizationHierarchy = await this.getOrganizationHierarchy(request.organizationId);

const userAttributes = await this.getUserAttributes(request.userId);

const resourceAttributes = await this.getResourceAttributes(request.resourceId);

const environmentAttributes = this.getEnvironmentAttributes(request);

return {

user: {

id: request.userId,

organizationId: request.organizationId,

primaryOrganizationId: user.primaryOrganizationId,

organizationMemberships: userOrganizations,

organizationHierarchy,

attributes: userAttributes,

},

resource: {

id: request.resourceId,

type: request.resourceType,

organizationId: request.resourceOrganizationId,

attributes: resourceAttributes,

},

environment: environmentAttributes,

action: request.action,

};

}

private async findHierarchicalPolicies(context: AuthorizationContext): Promise<Policy[]> {

// Get all ancestor and descendant organizations

const orgHierarchy = context.user.organizationHierarchy;

const relevantOrgIds = [

context.user.organizationId,

...orgHierarchy.ancestors,

...orgHierarchy.descendants,

];

const query = this.policyRepo

.createQueryBuilder('policy')

.leftJoinAndSelect('policy.rules', 'rules')

.leftJoinAndSelect('policy.organization', 'org')

.where('policy.isActive = true')

.andWhere(

'(policy.organizationId = :currentOrgId OR ' +

'(policy.organizationId IN (:...ancestorIds) AND policy.appliesTo Children = true) OR ' +

'(policy.organizationId IN (:...descendantIds) AND policy.inheritanceMode IN (:...inheritanceModes)))',

{

currentOrgId: context.user.organizationId,

ancestorIds: orgHierarchy.ancestors,

descendantIds: orgHierarchy.descendants,

inheritanceModes: ['inherit\_up', 'both'],

}

)

.orderBy('org.level', 'ASC') // Evaluate from root to leaf

.addOrderBy('policy.priority', 'DESC');

const policies = await query.getMany();

return policies.filter(policy =>

this.policyEvaluator.isApplicable(policy, context)

);

}

private async getUserOrganizations(userId: string): Promise<UserOrganizationMembership[]> {

return this.userOrgRepo

.createQueryBuilder('membership')

.leftJoinAndSelect('membership.organization', 'org')

.where('membership.userId = :userId', { userId })

.getMany();

}

private async getOrganizationHierarchy(organizationId: string): Promise<OrganizationHierarchy> {

const org = await this.orgRepo.findOne({

where: { id: organizationId },

relations: ['parent', 'children']

});

if (!org) {

throw new Error('Organization not found');

}

const ancestors = await this.getAncestors(organizationId);

const descendants = await this.getDescendants(organizationId);

return {

current: org,

ancestors: ancestors.map(a => a.id),

descendants: descendants.map(d => d.id),

level: org.level,

path: org.path,

};

}

}

// Enhanced Policy Evaluator with hierarchical conditions

@Injectable()

export class HierarchicalPolicyEvaluatorService extends PolicyEvaluatorService {

evaluateConditions(conditions: PolicyCondition[], context: AuthorizationContext): boolean {

if (!conditions || conditions.length === 0) return true;

return conditions.every(condition => {

switch (condition.operator) {

case 'equals':

return this.getAttributeValue(condition.attribute, context) === condition.value;

case 'in':

return condition.value.includes(this.getAttributeValue(condition.attribute, context));

case 'greater\_than':

return this.getAttributeValue(condition.attribute, context) > condition.value;

case 'regex':

return new RegExp(condition.value).test(this.getAttributeValue(condition.attribute, context));

case 'time\_in\_range':

return this.isTimeInRange(condition.value, new Date());

case 'organization\_level':

return this.evaluateOrganizationLevel(condition, context);

case 'organization\_type':

return this.evaluateOrganizationType(condition, context);

case 'in\_organization\_hierarchy':

return this.evaluateOrganizationHierarchy(condition, context);

case 'has\_role\_in\_organization':

return this.evaluateRoleInOrganization(condition, context);

default:

return false;

}

});

}

private evaluateOrganizationLevel(condition: PolicyCondition, context: AuthorizationContext): boolean {

const userOrgLevel = context.user.organizationHierarchy.level;

const requiredLevel = condition.value;

switch (condition.comparison || 'equals') {

case 'equals': return userOrgLevel === requiredLevel;

case 'greater\_than': return userOrgLevel > requiredLevel;

case 'less\_than': return userOrgLevel < requiredLevel;

case 'greater\_than\_or\_equal': return userOrgLevel >= requiredLevel;

case 'less\_than\_or\_equal': return userOrgLevel <= requiredLevel;

default: return false;

}

}

private evaluateOrganizationType(condition: PolicyCondition, context: AuthorizationContext): boolean {

const userOrgType = context.user.organizationHierarchy.current.organizationType;

return condition.value.includes(userOrgType);

}

private evaluateOrganizationHierarchy(condition: PolicyCondition, context: AuthorizationContext): boolean {

const targetOrgId = condition.value;

const userHierarchy = context.user.organizationHierarchy;

return userHierarchy.ancestors.includes(targetOrgId) ||

userHierarchy.descendants.includes(targetOrgId) ||

userHierarchy.current.id === targetOrgId;

}

private evaluateRoleInOrganization(condition: PolicyCondition, context: AuthorizationContext): boolean {

const { organizationId, role } = condition.value;

const membership = context.user.organizationMemberships.find(

m => m.organizationId === organizationId

);

return membership?.role === role;

}

}

## 

## 4. Admin Dashboard Requirements

### 4.1 Functional Requirements

#### Organization Management

* **Organization Hierarchy Management**
  + Create, edit, delete organizations with parent-child relationships
  + Visual tree view of organization hierarchy with drag-and-drop reordering
  + Organization type management (company, department, team, division)
  + Bulk organization operations (move, merge, archive)
  + Organization settings and configuration management

#### User Management

* **User CRUD Operations**
  + Create, read, update, delete users with organization assignments
  + Multi-organization membership management
  + Role assignment within organizations
  + User attribute management with inheritance visualization
  + Bulk user operations (import, export, bulk edit)
  + User search and filtering across organization hierarchy

#### ABAC Management

* **Policy Management**
  + Visual policy builder with drag-and-drop conditions
  + Policy templates library with parameterization
  + Policy inheritance configuration (applies to children, inheritance modes)
  + Policy testing sandbox with real-time evaluation
  + Policy versioning and rollback capabilities
  + Bulk policy operations
* **Attribute Management**
  + Attribute definition creation with data types and validation
  + Attribute inheritance configuration
  + Attribute value management for users and resources
  + Attribute template system for common patterns
* **Resource Management**
  + Business object CRUD (Products, Orders, Customers, Transactions)
  + Resource attribute management
  + Resource access control visualization
  + Resource sharing across organizations

#### Cross-Organization Features

* **Permission Management**
  + Grant/revoke cross-organization permissions
  + Time-bound permission management
  + Permission delegation workflows
  + Cross-organization audit trails

### 4.2 Non-Functional Requirements

#### Performance

* Support 10,000+ concurrent users across 1,000+ organizations
* Policy evaluation response time < 100ms for 95th percentile
* Dashboard page load time < 2 seconds
* Real-time updates using WebSocket connections
* Efficient caching with Redis for frequently accessed data

#### Scalability

* Horizontal scaling support for multiple admin instances
* Database sharding capability for large organization hierarchies
* Microservice architecture readiness
* CDN integration for static assets

#### Security

* Role-based access control for admin functions
* Audit logging for all administrative actions
* Data encryption at rest and in transit
* Rate limiting and DDoS protection
* OWASP security compliance

#### Usability

* Responsive design supporting mobile and tablet devices
* Internationalization (i18n) support for multiple languages
* Accessibility compliance (WCAG 2.1 AA)
* Comprehensive help system and documentation
* Keyboard navigation support

#### Technology Stack

* **Frontend**: React 18+ with NextJS 13+, TypeScript
* **UI Components**: SHADCN/UI with Radix UI primitives
* **Styling**: Tailwind CSS with custom design system
* **State Management**: Zustand for client state, React Query for server state
* **Forms**: React Hook Form with Zod validation
* **Charts**: Recharts for analytics and visualization
* **Icons**: Lucide React icon library

## 5. Demo Data and Example Implementation

### 5.1 Fictional Organizations

#### Organization 1: TechCorp Global

// Organization Structure

const techCorpStructure = {

id: 'org-1',

name: 'TechCorp Global',

type: 'company',

children: [

{

id: 'org-1-1',

name: 'Engineering Division',

type: 'division',

children: [

{ id: 'org-1-1-1', name: 'Backend Team', type: 'team' },

{ id: 'org-1-1-2', name: 'Frontend Team', type: 'team' },

{ id: 'org-1-1-3', name: 'DevOps Team', type: 'team' },

{ id: 'org-1-1-4', name: 'QA Team', type: 'team' }

]

},

{

id: 'org-1-2',

name: 'Sales Division',

type: 'division',

children: [

{ id: 'org-1-2-1', name: 'Enterprise Sales', type: 'team' },

{ id: 'org-1-2-2', name: 'SMB Sales', type: 'team' },

{ id: 'org-1-2-3', name: 'Customer Success', type: 'team' }

]

},

{

id: 'org-1-3',

name: 'Finance Department',

type: 'department',

children: [

{ id: 'org-1-3-1', name: 'Accounting', type: 'team' },

{ id: 'org-1-3-2', name: 'FP&A', type: 'team' }

]

}

]

};

// Sample Users

const techCorpUsers = [

{

id: 'user-1',

email: 'john.doe@techcorp.com',

firstName: 'John',

lastName: 'Doe',

jobTitle: 'Engineering Manager',

primaryOrganizationId: 'org-1-1',

organizationMemberships: [

{ organizationId: 'org-1', role: 'member' },

{ organizationId: 'org-1-1', role: 'manager' },

{ organizationId: 'org-1-1-1', role: 'admin' }

],

attributes: {

securityClearance: 'high',

department: 'engineering',

costCenter: 'ENG-001',

seniority: 'senior'

}

},

{

id: 'user-2',

email: 'jane.smith@techcorp.com',

firstName: 'Jane',

lastName: 'Smith',

jobTitle: 'Sales Director',

primaryOrganizationId: 'org-1-2',

organizationMemberships: [

{ organizationId: 'org-1', role: 'member' },

{ organizationId: 'org-1-2', role: 'admin' },

{ organizationId: 'org-1-2-1', role: 'manager' }

],

attributes: {

securityClearance: 'medium',

department: 'sales',

costCenter: 'SALES-001',

seniority: 'senior',

quota: 1000000

}

}

];

// Sample Products

const techCorpProducts = [

{

id: 'prod-1',

organizationId: 'org-1',

name: 'CloudSync Pro',

sku: 'CS-PRO-001',

price: 299.99,

category: 'Software',

status: 'active',

attributes: {

securityLevel: 'enterprise',

targetMarket: 'b2b',

supportTier: 'premium'

}

},

{

id: 'prod-2',

organizationId: 'org-1',

name: 'DataVault Basic',

sku: 'DV-BAS-001',

price: 99.99,

category: 'Software',

status: 'active',

attributes: {

securityLevel: 'standard',

targetMarket: 'smb',

supportTier: 'standard'

}

}

];

#### Organization 2: RetailMax Enterprises

// Organization Structure

const retailMaxStructure = {

id: 'org-2',

name: 'RetailMax Enterprises',

type: 'company',

children: [

{

id: 'org-2-1',

name: 'Operations Division',

type: 'division',

children: [

{ id: 'org-2-1-1', name: 'Store Operations', type: 'department' },

{ id: 'org-2-1-2', name: 'Supply Chain', type: 'department' },

{ id: 'org-2-1-3', name: 'Inventory Management', type: 'department' }

]

},

{

id: 'org-2-2',

name: 'Regional Stores',

type: 'division',

children: [

{ id: 'org-2-2-1', name: 'North Region', type: 'region' },

{ id: 'org-2-2-2', name: 'South Region', type: 'region' },

{ id: 'org-2-2-3', name: 'East Region', type: 'region' },

{ id: 'org-2-2-4', name: 'West Region', type: 'region' }

]

},

{

id: 'org-2-3',

name: 'Corporate Services',

type: 'division',

children: [

{ id: 'org-2-3-1', name: 'HR Department', type: 'department' },

{ id: 'org-2-3-2', name: 'IT Department', type: 'department' },

{ id: 'org-2-3-3', name: 'Marketing Department', type: 'department' }

]

}

]

};

// Sample Users

const retailMaxUsers = [

{

id: 'user-3',

email: 'bob.wilson@retailmax.com',

firstName: 'Bob',

lastName: 'Wilson',

jobTitle: 'Regional Manager',

primaryOrganizationId: 'org-2-2-1',

organizationMemberships: [

{ organizationId: 'org-2', role: 'member' },

{ organizationId: 'org-2-2', role: 'member' },

{ organizationId: 'org-2-2-1', role: 'admin' }

],

attributes: {

accessLevel: 'regional',

department: 'operations',

region: 'north',

managerLevel: 2

}

},

{

id: 'user-4',

email: 'alice.brown@retailmax.com',

firstName: 'Alice',

lastName: 'Brown',

jobTitle: 'Store Manager',

primaryOrganizationId: 'org-2-2-1',

organizationMemberships: [

{ organizationId: 'org-2', role: 'member' },

{ organizationId: 'org-2-2-1', role: 'member' }

],

attributes: {

accessLevel: 'store',

department: 'operations',

region: 'north',

storeId: 'store-001',

managerLevel: 1

}

}

];

// Sample Products

const retailMaxProducts = [

{

id: 'prod-3',

organizationId: 'org-2',

name: 'Premium Coffee Beans',

sku: 'PCB-001',

price: 24.99,

category: 'Food & Beverage',

status: 'active',

attributes: {

perishable: true,

supplier: 'GlobalCoffee Inc',

margin: 0.35,

seasonality: 'none'

}

},

{

id: 'prod-4',

organizationId: 'org-2',

name: 'Wireless Headphones',

sku: 'WH-PRO-001',

price: 149.99,

category: 'Electronics',

status: 'active',

attributes: {

perishable: false,

supplier: 'TechSupplier Ltd',

margin: 0.50,

seasonality: 'holiday'

}

}

];

### 5.2 Example Policy Implementations

#### TechCorp Policies

const techCorpPolicies = [

{

id: 'policy-tc-1',

name: 'Engineering Read Access',

organizationId: 'org-1-1', // Engineering Division

appliesToChildren: true,

effect: 'permit',

target: {

resourceType: 'product',

action: 'read'

},

conditions: [

{

attribute: 'user.department',

operator: 'equals',

value: 'engineering'

}

],

priority: 100

},

{

id: 'policy-tc-2',

name: 'Senior Engineer Product Management',

organizationId: 'org-1-1',

appliesToChildren: true,

effect: 'permit',

target: {

resourceType: 'product',

action: ['create', 'update']

},

conditions: [

{

attribute: 'user.department',

operator: 'equals',

value: 'engineering'

},

{

attribute: 'user.seniority',

operator: 'in',

value: ['senior', 'principal', 'staff']

},

{

attribute: 'user.securityClearance',

operator: 'in',

value: ['medium', 'high']

}

],

priority: 200

},

{

id: 'policy-tc-3',

name: 'Sales Team Customer Access',

organizationId: 'org-1-2', // Sales Division

appliesToChildren: true,

effect: 'permit',

target: {

resourceType: 'customer',

action: ['read', 'update']

},

conditions: [

{

attribute: 'user.department',

operator: 'equals',

value: 'sales'

},

{

attribute: 'resource.assignedTo',

operator: 'equals',

value: '${user.id}' // Dynamic reference

}

],

priority: 150

},

{

id: 'policy-tc-4',

name: 'High Value Order Approval',

organizationId: 'org-1',

appliesToChildren: true,

effect: 'deny',

target: {

resourceType: 'order',

action: 'approve'

},

conditions: [

{

attribute: 'resource.totalAmount',

operator: 'greater\_than',

value: 50000

},

{

attribute: 'user.organizationLevel',

operator: 'greater\_than',

value: 2 // Only level 2+ can approve

}

],

priority: 300

},

{

id: 'policy-tc-5',

name: 'Finance Department Transaction Access',

organizationId: 'org-1-3', // Finance Department

appliesToChildren: true,

effect: 'permit',

target: {

resourceType: 'transaction',

action: ['read', 'create', 'update']

},

conditions: [

{

attribute: 'user.department',

operator: 'equals',

value: 'finance'

}

],

priority: 180

}

];

#### RetailMax Policies

const retailMaxPolicies = [

{

id: 'policy-rm-1',

name: 'Regional Manager Full Access',

organizationId: 'org-2-2', // Regional Stores

appliesToChildren: true,

effect: 'permit',

target: {

resourceType: ['product', 'customer', 'order', 'transaction'],

action: ['read', 'create', 'update']

},

conditions: [

{

attribute: 'user.jobTitle',

operator: 'equals',

value: 'Regional Manager'

},

{

attribute: 'user.accessLevel',

operator: 'in',

value: ['regional', 'corporate']

}

],

priority: 200

},

{

id: 'policy-rm-2',

name: 'Store Manager Limited Access',

organizationId: 'org-2-2',

appliesToChildren: true,

effect: 'permit',

target: {

resourceType: ['product', 'customer', 'order'],

action: ['read', 'update']

},

conditions: [

{

attribute: 'user.jobTitle',

operator: 'equals',

value: 'Store Manager'

},

{

attribute: 'user.region',

operator: 'equals',

value: '${resource.region}' // Can only access resources in their region

}

],

priority: 150

},

{

id: 'policy-rm-3',

name: 'High Value Transaction Restriction',

organizationId: 'org-2',

appliesToChildren: true,

effect: 'deny',

target: {

resourceType: 'transaction',

action: 'create'

},

conditions: [

{

attribute: 'resource.amount',

operator: 'greater\_than',

value: 10000

},

{

attribute: 'user.managerLevel',

operator: 'less\_than',

value: 2

}

],

priority: 300

},

{

id: 'policy-rm-4',

name: 'Perishable Product Management',

organizationId: 'org-2-1-2', // Supply Chain

appliesToChildren: false,

effect: 'permit',

target: {

resourceType: 'product',

action: ['create', 'update', 'delete']

},

conditions: [

{

attribute: 'resource.perishable',

operator: 'equals',

value: true

},

{

attribute: 'user.department',

operator: 'equals',

value: 'operations'

},

{

attribute: 'environment.time',

operator: 'time\_in\_range',

value: { start: '06:00', end: '18:00' } // Business hours only

}

],

priority: 250

}

];



# Instruction

1. I want to develop this platform and test it
2. Create all the needed resources for building this project with claude-code
3. Use this document for detailed and complementary information
4. Create all the [todo.md](http://todo.md) tasks very detailed and follow the instructions and information in this document
5. Make sure to referenced all the resources in the [claude.md](http://claude.md) file.
6. I will add this document to the resources folder in the project, reference this document also in the [claude.md](http://claude.md) file.
7. Create a developer guide to build this project. Make sure to write the actions the developer need to do or take assuming using claude-code and these instructions and resources, and not general development guide and tasks. Just the necessary minimal actions.
8. Prepare the resources for developing with claude-code and vs-code. Create only the resources needed for development in vs-code IDE. Leave the other arifcats to be created by claude-code while developing the entire project. For example, don’t create the .env, or docker-compose.yaml files, they will be created with claude-code, based on the [claude.md](http://claude.md), [todo.md](http://todo.md) and all the other resources that you should create. Make sure we have all the necessary resources and instructions for building the project exactly as described.