STS TaxRepair – Branding & Theme Permission Specification

This document defines the branding-related permissions, database structure, API behavior, and audit logging for the STS TaxRepair multi-office system. It is focused ONLY on branding, themes, and related access control.

1) New Permissions to Add (RBAC)

New Permissions

Add these permissions to your permissions table (or equivalent):

• manage_office_branding

• manage_personal_theme

• manage_system_branding (Admin-only, global)

Role → Permission Mapping

Roles: Client, Agent, TaxOffice, Admin

Permission: manage_office_branding

- Client: NO

- Agent: NO

- TaxOffice: YES (office-scoped)

- Admin: YES (global)

Permission: manage_personal_theme

- Client: OPTIONAL (OK to allow)

- Agent: YES (personal only)

- TaxOffice: YES (personal only)

- Admin: YES (personal only)

Permission: manage_system_branding

- Client: NO

- Agent: NO

- TaxOffice: NO

- Admin: YES (global STS-level branding only)

Important:

manage_office_branding → TaxOffice can only update branding for their own office_id

manage_system_branding → Admin can update STS TaxRepair global branding (default logo, global theme, etc.)

2) Database Structure for Branding

Assuming you already have an offices table and a users table with office_id for Tax Office and Agent users.

a) System-Level Branding (STS TaxRepair)

Create a table like:

```
CREATE TABLE system_branding (
    id               INT PRIMARY KEY AUTO_INCREMENT,
    logo_url         VARCHAR(255) NULL,
    primary_color    VARCHAR(20)  NULL,
    secondary_color  VARCHAR(20)  NULL,
    accent_color     VARCHAR(20)  NULL,
    default_theme    ENUM('light', 'dark') DEFAULT 'light',
    updated_by_user_id INT NULL,
    updated_at       TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

Only Admin can update this (via manage_system_branding). This acts as your global fallback if office branding is not set.

b) Office-Level Branding

Create an office_branding table (or extend offices if you prefer; separate table is cleaner):

```
CREATE TABLE office_branding (
    id               INT PRIMARY KEY AUTO_INCREMENT,
    office_id        INT NOT NULL,
    logo_url         VARCHAR(255) NULL,
    primary_color    VARCHAR(20)  NULL,
    secondary_color  VARCHAR(20)  NULL,
    accent_color     VARCHAR(20)  NULL,
    default_theme    ENUM('light', 'dark') DEFAULT 'light',
    updated_by_user_id INT NULL,
    updated_at       TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    CONSTRAINT fk_office_branding_office
        FOREIGN KEY (office_id) REFERENCES offices(id)
        ON DELETE CASCADE
);
```

Rules:

• Each office can have 0 or 1 row in office_branding.

• If no row exists → use system_branding as fallback.

• updated_by_user_id references users.id.

c) Personal Theme Preferences (Per User)

Add a small preference to users or a separate user_preferences table. Simplest approach: extend users:

```
ALTER TABLE users
ADD COLUMN theme_preference ENUM('system', 'light', 'dark') DEFAULT 'system';
```

This is controlled by manage_personal_theme. It only affects that user's UI, not branding.

3) API / Application Behavior

a) Resolving Branding for a Request

When rendering the app (web dashboard, client portal, etc.), the branding resolution should follow:

```
For any authenticated request:

if user.role == 'Admin':
    // Use system_branding for admin views
    branding = system_branding

else:
    // For TaxOffice, Agent, Client:
    officeId = user.office_id

    officeBranding = SELECT * FROM office_branding WHERE office_id = officeId LIMIT 1

    if officeBranding exists:
        branding = officeBranding
    else:
        branding = system_branding
```

Key behavior:

• Tax Office’s logo & theme affect their office’s agents and clients.

• Other offices are unaffected.

• If an office deletes or never sets branding → the system falls back to STS defaults.

b) Who Can Update What

System Branding (STS-Level):

Endpoint example: PUT /api/system/branding

Only allowed if:

```
user.role == 'Admin' AND
User has manage_system_branding permission
```

Office Branding:

Endpoint example: PUT /api/offices/:officeId/branding

Allowed if:

```
User has manage_office_branding, AND
One of:
    user.role == 'TaxOffice' AND user.office_id == officeId
    user.role == 'Admin' (Admin can override any office)
```

Personal Theme:

Endpoint example: PUT /api/users/me/preferences

Only touches theme_preference field on the current user row.

Allowed if user has manage_personal_theme.

4) Audit Logging (Important)

Any change to branding should be auditable. Create an audit_logs entry on these actions:

• System branding update

• Office branding update

• (Optional) Personal theme change

Example structure:

```
CREATE TABLE audit_logs (
    id          INT PRIMARY KEY AUTO_INCREMENT,
    user_id     INT NOT NULL,
    action      VARCHAR(100) NOT NULL,
    entity_type VARCHAR(50) NOT NULL,
    entity_id   INT NULL,
    metadata    JSON NULL,
    created_at  TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

When office branding changes, insert something like:

```
{
  "old_values": {
    "logo_url": "https://old-logo.png",
    "primary_color": "#0f766e"
  },
  "new_values": {
    "logo_url": "https://new-logo.png",
    "primary_color": "#1d4ed8"
  }
}
```

Visibility:

• Tax Office can view logs for: entity_type = 'office_branding' with their office_id

• Admin can view all logs.

■ TL;DR for the Dev

Key concepts to implement:

• Add permission: manage_office_branding, manage_personal_theme, manage_system_branding.

• Tax Office + Admin can manage office branding (office-scoped); only Admin manages global branding.

• Add system_branding, office_branding, and optionally theme_preference on users.

• Apply branding resolution: office_branding → fallback to system_branding.

• Scope updates by office_id for Tax Office users.

• Log all branding changes in audit_logs.