Sundaram Portal - Notification System Documentation

Overview

The notification system in Sundaram Portal is role-based and handles different types of notifications for administrators, checkers, and makers. The system tracks changes, approvals, and rejections across different tables.

Database Structure

The system primarily uses the `app.change_tracker` table to manage notifications and track changes.

Role-Based Features

1. Admin Notifications

API Endpoint: `getAdminNotification`

// GET /api/admin/notifications

Features:

- View pending rows grouped by table
- See count of pending requests per table
- Track changes awaiting approval
- Monitor system-wide changes

Response Structure:

```
{
  "success": true,
  "data": [
    {
      "table_name": "string",
      "pending_count": number,
      "latest_changes": [
        {
          "request_id": "uuid",
          "row_id": "string",
          "status": "string",
```

```
        "created_at": "timestamp"
      }
    ]
  }
 ]
}
```

2. Checker Notifications

API Endpoint: `getCheckerNotification`

```
// GET /api/checker/notifications
```

Features:

- View pending requests assigned to them
- Track changes requiring their review
- See request details including:
  - Table name
  - Row ID
  - Request ID
  - Creation timestamp
  - Status

Response Structure:

```
{
  "success": true,
  "data": [
    {
      "request_id": "uuid",
      "table_name": "string",
      "row_id": "string",
      "status": "string",
```

```
    "created_at": "timestamp"
   }
 ]
}
```


Notification Triggers

1. Row Addition

When a maker adds a new row:

- Creates entry in change_tracker
- Status set to "pending"
- Notifies relevant checkers

2. Row Updates

When changes are made:

- Updates change_tracker entry
- Status changes to "pending"
- Notifies checkers for review

3. Approval/Rejection

When checker takes action:

- Updates change_tracker status
- Notifies admin of decision
- Updates original table if approved

Status Types

1. `pending` - Awaiting checker review
2. `approved` - Changes accepted
3. `rejected` - Changes declined
4. `admin_approved` - Final admin approval

Error Handling

The system includes comprehensive error handling:

```
try {
  // Operation logic
} catch (error) {
  return res.status(500).json({
    success: false,
    message: "An error occurred while processing the request",
    error: error.message
  });
}
```

Security Features

1. JWT-based authentication
2. Role-based access control
3. Input sanitization
4. Transaction management for data integrity

Database Queries

Example of notification query:

```
SELECT
    table_name,
    COUNT(*) as pending_count,
    json_agg(json_build_object(
        'request_id', request_id,
        'row_id', row_id,
        'status', status,
        'created_at', created_at
    )) as latest_changes
```

```
FROM app.change_tracker
WHERE status = 'pending'
GROUP BY table_name
ORDER BY pending_count DESC;
```

Best Practices

1. Always use transactions for data consistency
2. Implement proper error handling
3. Sanitize user inputs
4. Use parameterized queries
5. Maintain audit trails

Integration Points

1. Frontend notification components
2. Audit logging system