Functional Specification Document: Retail Execution Audit System

1. Introduction
This document outlines the functional specifications for the **Retail Execution Audit System**, including template creation, audit execution, and monitoring capabilities. It is designed to allow FMCG companies to create, configure, deploy, and monitor audits dynamically, ensuring operational efficiency and compliance.

1.1 Objective
The goal of this system is to provide users with an intuitive and flexible platform to design customizable audits, execute them in the field, and analyze results for actionable insights.

---

2. Functional Requirements

2.1 Template Creation Workflow

The template creation process follows a step-by-step wizard approach. Each step provides specific functionality:

Step 1: Template Setup
- **Inputs**:
  - Template Name (Text Input): Unique identifier for the template.
  - Template Description (Text Area): Brief description of the template.
  - Audit Category (Dropdown): Predefined categories such as Merchandising, Stock, Quality, etc.
- **Actions**:
  - "Next" button: Proceeds to the next step.
  - "Cancel" button: Exits the wizard.
- **Validation**:
  - Template Name: Mandatory field.

Step 2: Define Sections
- **Inputs**:
  - Add Section Button:
    - Opens a modal to enter:
      - Section Title (Text Input): Name of the section.
      - Section Description (Optional Text Area): Brief explanation.
  - Drag-and-Drop Reordering: Allows rearranging sections.
- **Actions**:
  - "Add Section" button: Adds a new section.
  - "Edit" and "Delete" icons: Modify or remove sections.
  - "Previous" and "Next" buttons: Navigate between steps.
  - "Cancel" button: Exits the wizard.
- **Validation**:
  - Section Title: Mandatory field.

Step 3: Add Questions
- **Inputs**:
  - Dropdown: Select the section where the question will be added.
  - Question Type Dropdown:
    - Options: Text Input, Numeric Input, Single Choice, Multiple Choice, Dropdown, Date/Time, File Upload, Barcode Scanner.
  - Question Text (Text Input): Define the question.
  - Options for applicable question types (e.g., Single Choice, Multiple Choice): Add predefined answers.
  - Validation Rules:
    - Mandatory: Checkbox to make the question required.
    - Numeric Range: Min/Max values for numeric inputs.
- **Actions**:
  - "Add Question" button: Opens a modal to configure a new question.
  - "Preview Question" button: Provides a real-time view of the question.
  - "Previous" and "Next" buttons: Navigate between steps.
  - "Cancel" button: Exits the wizard.
- **Validation**:
  - Question Text: Mandatory field.

Step 4: Configure Logic
- **Inputs**:
  - Dropdown: Select a question to configure logic.
  - Conditional Logic Builder:
    - Define conditions using operators (AND, OR, NOT).
    - Specify actions (e.g., "Show Question 5" or "Skip to Section 3").
  - "Test Logic" button: Simulate responses to verify conditional flows.
- **Actions**:
  - "Add Condition" button: Adds new logic rules.
  - "Previous" and "Next" buttons: Navigate between steps.
  - "Cancel" button: Exits the wizard.
- **Validation**:
  - Ensure logic does not create circular dependencies or unreachable paths.

Step 5: Scoring and Publish
- **Inputs**:
  - Scoring Toggle: Enable or disable scoring for the template.
  - Weights Assignment:
    - Table to assign weights to sections or questions.
    - Critical Questions Checkbox: Mark specific questions as critical.
  - Compliance Threshold (Numeric Input): Define pass/fail criteria.
- **Actions**:
  - "Save as Draft" button: Saves the template for later editing.
  - "Publish Template" button: Finalizes and deploys the template.
  - "Previous" button: Navigate to the previous step.
- **Validation**:

  - Ensure total weights sum up to 100% if scoring is enabled.

2.2 Audit Execution Workflow

Step 1: Select an Audit
- **Inputs**:
  - Dropdown: Select an assigned audit from the list.
  - Filters: Filter audits by date, region, or category.
- **Actions**:
  - "Start Audit" button: Begins the audit execution process.

Step 2: Execute Audit
- **Inputs**:
  - Navigate through the sections of the audit.
  - Answer questions based on template configurations (e.g., text input, multiple choice).
  - Upload photos or files for evidence.
  - Use barcode/QR code scanner for product validation.
- **Actions**:
  - "Save Progress" button: Saves the current state for later completion.
  - "Submit Audit" button: Finalizes and submits the audit.

Step 3: Sync Data
- **Inputs**:
  - Offline Mode: Enable audits to be completed offline.
  - Automatic Sync: Sync data with the server when online.
- **Actions**:
  - "Sync Now" button: Manually triggers data synchronization.

2.3 Monitoring and Reporting

Dashboard
- **Features**:
  - Real-time metrics for audit completion rates, compliance scores, and issue trends.
  - Filter audits by region, category, or personnel.

Reports
- **Outputs**:
  - Exportable formats (PDF, Excel, CSV).
  - Customizable reports to include specific fields, scores, and timestamps.

Notifications
- **Triggers**:
  - Alerts for overdue audits or critical compliance failures.

  - Notifications for new audit assignments.

---

3. Non-Functional Requirements

3.1 Usability
- Provide a clean, intuitive interface with clear labels and tooltips.

3.2 Performance
- Support up to 500 simultaneous users across all modules.

3.3 Security
- Role-based access control for all features.
- Encrypt sensitive data at rest and in transit.

---

4. System Design Overview

4.1 Architecture
- **Frontend**: Built using Blazor for web and mobile compatibility.
- **Backend**: .NET Core APIs for business logic and data handling.
- **Database**: PostgreSQL with JSONB columns to store dynamic template structures.

4.2 Data Model
- **Templates Table**:
  - TemplateID (Primary Key)
  - Name
  - Description
  - Category
  - Sections (JSONB)
  - Scoring Rules (JSONB)
  - PublishedStatus (Boolean)

- **Audits Table**:
  - AuditID (Primary Key)
  - TemplateID (Foreign Key)
  - Status (In Progress, Completed)
  - AssignedTo (User ID)
  - Responses (JSONB)
  - SubmittedAt (Timestamp)

- **Users Table**:
  - UserID (Primary Key)
  - Name
  - Role

- AssignedRegions

---

5. Deliverables
- Fully functional template creation, audit execution, and reporting modules.
- Mobile-friendly audit execution with offline capabilities.
- Dynamic dashboards and exportable reports.

---

6. Acceptance Criteria
1. Users can create, edit, and publish templates.
2. Field users can execute audits and sync data offline.
3. Dashboards display real-time insights with accurate filtering.
4. Reports are customizable and exportable in multiple formats.

---

7. Appendix
- Sample audit templates for reference.
- UI mockups for each module.
- Detailed API specifications for integration.

Database Design Document: Retail Execution Audit System

1. Introduction
This document provides the database design for the Retail Execution Audit System, including schema definitions, data storage strategies, and examples of JSON-based storage and retrieval for audit results. It also includes examples of scoring logic and report generation.

---

2. Database Schema

2.1 Templates Table
- **Purpose**: Stores audit template metadata and configuration.
- **Columns**:
  - `template_id` (UUID, Primary Key): Unique identifier for each template.
  - `name` (VARCHAR, Not Null): Name of the template.
  - `description` (TEXT): Brief description of the template.
  - `category` (VARCHAR): Category of the audit (e.g., Merchandising, Stock, Quality).
  - `sections` (JSONB): Stores the hierarchical structure of sections and questions.
  - `scoring_rules` (JSONB): Configuration for scoring and compliance thresholds.
  - `created_by` (UUID, Foreign Key to `users.user_id`): The user who created the template.

  - `created_at` (TIMESTAMP, Default: `CURRENT_TIMESTAMP`): Timestamp of template creation.
  - `updated_at` (TIMESTAMP, Default: `CURRENT_TIMESTAMP`): Timestamp of last update.
  - `is_published` (BOOLEAN, Default: FALSE): Indicates whether the template is published.

2.2 Audits Table
- **Purpose**: Tracks individual audit instances created from templates.
- **Columns**:
  - `audit_id` (UUID, Primary Key): Unique identifier for each audit.
  - `template_id` (UUID, Foreign Key to `templates.template_id`): Links to the template used.
  - `status` (VARCHAR, Default: 'Pending'): Status of the audit (Pending, In Progress, Completed).
  - `assigned_to` (UUID, Foreign Key to `users.user_id`): User assigned to the audit.
  - `location` (JSONB): Stores location details (e.g., store name, GPS coordinates).
  - `responses` (JSONB): Captures responses for all sections and questions.
  - `score` (NUMERIC): Calculated score based on the template's scoring rules.
  - `submitted_at` (TIMESTAMP): Timestamp of audit submission.
  - `created_at` (TIMESTAMP, Default: `CURRENT_TIMESTAMP`): Timestamp of audit creation.

2.3 Sections Table
- **Purpose**: Stores predefined reusable sections for templates.
- **Columns**:
  - `section_id` (UUID, Primary Key): Unique identifier for each section.
  - `template_id` (UUID, Foreign Key to `templates.template_id`): Links to the parent template.
  - `title` (VARCHAR, Not Null): Title of the section.
  - `description` (TEXT): Description of the section.
  - `order` (INTEGER): Sequence of the section within the template.
  - `questions` (JSONB): Stores all questions and their configurations for the section.

2.4 Users Table
- **Purpose**: Manages user accounts and roles.
- **Columns**:
  - `user_id` (UUID, Primary Key): Unique identifier for each user.
  - `name` (VARCHAR, Not Null): Full name of the user.
  - `email` (VARCHAR, Not Null, Unique): User's email address.
  - `role` (VARCHAR, Not Null): User role (e.g., Admin, Auditor, Supervisor).
  - `assigned_regions` (JSONB): Regions or territories assigned to the user.
  - `password_hash` (TEXT, Not Null): Hashed password for authentication.

- `created_at` (TIMESTAMP, Default: `CURRENT_TIMESTAMP`): Timestamp of account creation.
- `last_login` (TIMESTAMP): Timestamp of the user's last login.

2.5 Reports Table
- **Purpose**: Stores generated reports for audits and templates.
- **Columns**:
- `report_id` (UUID, Primary Key): Unique identifier for each report.
- `audit_id` (UUID, Foreign Key to `audits.audit_id`): Links to the audit.
- `generated_by` (UUID, Foreign Key to `users.user_id`): User who generated the report.
- `data` (JSONB): Contains the raw data of the report.
- `created_at` (TIMESTAMP, Default: `CURRENT_TIMESTAMP`): Timestamp of report creation.

---

3. Audit Result Storage Example

3.1 JSON Structure for Audit Responses
- **Template Configuration (Example)**:
```json
{
  "sections": [
    {
      "section_id": "section_1",
      "title": "Shelf Compliance",
      "questions": [
        {
          "question_id": "q1",
          "text": "Is the shelf organized as per planogram?",
          "type": "single_choice",
          "options": ["Yes", "No"]
        },
        {
          "question_id": "q2",
          "text": "Upload a photo of the shelf.",
          "type": "file_upload"
        }
      ]
    }
  ]
}
```

- **Audit Responses (Example)**:

```json
{
  "section_1": {
    "q1": "Yes",
    "q2": "file://path/to/photo.jpg"
  }
}
```

3.2 Query to Retrieve Audit Results
Retrieve responses for a specific audit:
```sql
SELECT responses
FROM audits
WHERE audit_id = 'AUDIT_UUID';
```

Output Example:
```json
{
  "section_1": {
    "q1": "Yes",
    "q2": "file://path/to/photo.jpg"
  }
}
```

---

4. Scoring Logic

4.1 Scoring Configuration (Template)
```json
{
  "scoring_rules": {
    "weights": {
      "section_1": 50,
      "section_2": 50
    },
    "threshold": 80,
    "critical_questions": ["q1"]
  }
}
```

4.2 Scoring Calculation
- Assign scores for each question:

  - `q1`: Yes = 10 points, No = 0 points.
  - `q2`: File upload = 10 points.
- Calculate total score:
  - Section 1 Total: `(q1_score + q2_score) / Max Points * Section Weight`.
  - Overall Total: Sum of all section scores.

Example Calculation:
- Responses:
  - `q1`: Yes → 10 points.
  - `q2`: File uploaded → 10 points.
- Scoring:
  - Section 1 Score: `(10 + 10) / 20 * 50 = 50`.
  - Total Score: 50 (Section 1).
- Compliance Check:
  - Threshold: 80.
  - Result: Non-Compliant.

---

5. Reporting Examples

5.1 Generate Compliance Report
Retrieve audits below compliance threshold:
```sql
SELECT a.audit_id, a.score, t.name AS template_name, u.name AS auditor_name
FROM audits a
JOIN templates t ON a.template_id = t.template_id
JOIN users u ON a.assigned_to = u.user_id
WHERE a.score < (t.scoring_rules->>'threshold')::NUMERIC;
```

Output Example:
audit_id	score	template_name	auditor_name
AUDIT12345	70	Shelf Compliance	John Doe

5.2 Generate Section-Wise Breakdown
Retrieve scores per section for a specific audit:
```sql
SELECT responses->'section_1' AS section_1_responses,
     responses->'section_2' AS section_2_responses
FROM audits
WHERE audit_id = 'AUDIT_UUID';
```

---

6. Indexing Recommendations
1. `templates (is_published)`: For quick access to published templates.
2. `audits (assigned_to, status)`: For filtering user-specific audits.
3. `users (email)`: For login authentication.

This design ensures that the system can efficiently handle dynamic data, scalable queries, and accurate reporting. Let me know if further refinements are needed.