# Feeder Performance Report System - Detailed Explanation

## Overview

This system generates Excel-based performance reports for electrical energy feeders, tracking their energy consumption, nominations, and compliance with various performance metrics. It's designed for utility companies or energy management systems to monitor feeder performance across different regions and business hubs.

## Core Components

### 1. Data Models & Interfaces

- **Feeder**: Represents electrical feeders with properties like name, business hub, region, energy uptake capacity
- **FeederReading**: Daily energy consumption readings with cumulative energy consumption data
- **Region & BusinessHub**: Geographic and organizational groupings for feeders
- **PopulatedFeeder**: Extended feeder interface with populated region and business hub names

### 2. Key Interfaces

```typescript
interface IReportParams {
  region?: string;
  businessHub?: string;
  dateRange: IReportDateRange;
}

interface FailedChecks {
  feederName: string;
  businessHub: string;
  region: string;
  date: string;
  failedChecks: string[];
}
```

## Service Classes

### FeederDataService

Handles data fetching and caching operations:

- **Caching System**: Implements a 5-minute TTL cache for feeder readings to improve performance
- **Data Fetching**: Retrieves feeders with populated region/business hub information
- **Reading Aggregation**: Groups feeder readings by feeder ID for efficient processing
- **Database Queries**: Optimized queries with proper sorting and filtering

### ExcelService

Manages Excel workbook operations:

- **Template Loading**: Loads a predefined Excel template for consistent formatting

- **Dynamic Headers**: Creates date-based column headers with sub-columns for Nomination, Actual, and Variance
- **Worksheet Management**: Creates multiple analysis sheets for different compliance categories
- **Styling**: Applies borders, colors, fonts, and cell formatting

### ReportGenerationService

Core business logic for report generation:

- **Data Population**: Fills Excel sheets with feeder performance data
- **Compliance Checking**: Runs various performance checks against business rules
- **Regional Grouping**: Organizes feeders by region in the report
- **Variance Calculations**: Computes differences between actual and nominated values

## Key Features

### Performance Metrics Tracked

1. **Daily Energy Nomination** vs **Actual Consumption**
2. **Cumulative Energy Consumption** over time periods
3. **Variance Analysis** (positive/negative deviations)
4. **Daily Energy Uptake** compliance
5. **Monthly Delivery Plan** tracking

### Compliance Checks

The system performs several automated compliance checks:

- **Actual D-0 < Actual D-1**: Ensures current day consumption isn't less than previous day
- **< 70% Nomination**: Flags feeders consuming less than 70% of nominated energy

- **> 130% Nomination**: Flags feeders exceeding 130% of nominated energy
- **< 70% Daily Uptake**: Daily consumption below 70% of expected uptake
- **> 130% Daily Uptake**: Daily consumption exceeding 130% of expected uptake
- **Positive Variance**: Tracks positive deviations from plan

#### Analysis Sheets

Creates separate Excel sheets for each compliance category:

- Failed compliance checks by category
- Summary of all issues
- Clean data for feeders with no flags

## API Endpoints

### 1. Generate Daily All Feeders Report

```
typescript

generateDailyAllFeedersReportBuffer(req, res): Promise<Buffer>
```

- Creates comprehensive reports for all feeders
- Includes compliance analysis
- Returns Excel buffer for download or email

### 2. Generate Specific Date Report

```
typescript

generateSpecificDateReport(req, res): Promise<void>
```

- Generates report for a specific date
- Supports filtering by region and business hub

- Downloads directly as Excel file

### 3. Generate Feeder-Specific Report

typescript

```
generateFeederSpecificReport(req, res): Promise<void>
```

- Creates reports for selected feeders by ID
- Supports date range selection
- Useful for detailed analysis of specific feeders

### 4. Send Report by Email

typescript

```
sendReportByEmail(req, res): Promise<void>
```

- Generates and emails reports automatically
- Supports various filtering options
- Includes email attachment functionality

### 5. Generate Custom Report

typescript

```
generateCustomReport(req, res): Promise<void>
```

- Flexible report generation with multiple parameters
- Supports different report types
- Configurable analysis inclusion

## Data Processing Flow

### 1. Data Preparation

- Fetch feeders from database with populated references
- Retrieve feeder readings for specified date range
- Group readings by feeder for efficient processing
- Apply regional/business hub filters if specified

### 2. Excel Generation

- Load predefined template
- Setup dynamic date headers based on report period
- Create analysis worksheets for compliance categories
- Apply consistent formatting and styling

### 3. Data Population

- Organize feeders by region for structured presentation
- Calculate nominations based on daily energy uptake
- Populate actual consumption from readings
- Compute variances and apply color coding
- Run compliance checks and categorize issues

### 4. Output Generation

- Generate Excel buffer for download
- Apply final formatting and styling
- Create downloadable file or email attachment

## Utility Functions

### Date Handling

- `getDatesInRange()`: Generates array of dates between start and end
- `formatDate()`: Consistent date formatting across the system
- `calculateDateRange()`: Determines date ranges for reports

### Excel Utilities

- `getColumnLetter()`: Converts numeric column index to Excel letter notation
- `createBorderStyle()`: Consistent border styling for cells
- Column width management and cell formatting

### Compliance Logic

- Automated performance threshold checking
- Previous day comparison logic
- Percentage-based compliance calculations
- Failed checks categorization and reporting

## Performance Optimizations

### Caching Strategy

- 5-minute TTL cache for feeder readings
- Reduces database queries for frequently accessed data
- Improves response times for report generation

### Batch Processing

- Parallel data fetching using Promise.all()
- Efficient database queries with proper indexing
- Optimized Excel generation with minimal memory usage

### Error Handling

- Comprehensive try-catch blocks
- Proper HTTP status codes
- Detailed error messages for debugging
- Graceful degradation for missing data

## Use Cases

### Daily Operations

- Monitor feeder performance across regions
- Identify underperforming or overperforming feeders
- Track compliance with energy delivery plans
- Generate automated daily reports

### Analysis & Reporting

- Historical performance analysis
- Compliance trend monitoring
- Regional performance comparisons
- Custom report generation for stakeholders

### Alerting & Notifications

- Email-based report distribution
- Automated compliance alerts
- Performance threshold notifications
- Executive summary reports

## Technical Architecture

### Database Integration

- MongoDB with Mongoose ODM
- Populated references for efficient querying
- Proper indexing for performance
- Lean queries for memory optimization

### Excel Generation

- ExcelJS library for Excel manipulation
- Template-based approach for consistency
- Dynamic content generation
- Advanced formatting and styling

### Email Integration

- Automated email delivery system
- Attachment handling for Excel files
- Configurable recipients and content
- Error handling for delivery failures

This system provides a comprehensive solution for energy feeder performance monitoring, combining data analysis, compliance checking, and automated reporting in a scalable, maintainable architecture.