4-Views Model for the SmartHoover Performance System

---

1. Context View

The context view shows the external perspective of the SmartHoover Performance System, including all actors and external systems that interact with it.

1.1 Actors

- **Salesman**
  A SmartHoover employee who can view his own performance records.
- **Supervisor**
  Responsible for creating, updating, reviewing, and approving performance records of assigned salesmen.
- **HR Manager**
  Uses the system to view HR-related information from OrangeHRM and to support performance evaluations.
- **System Administrator**
  Responsible for user management, configuration, and operational monitoring.

1.2 External Systems

- **OrangeHRM**
  External Human Resource Management System.
  Provides employee master data (e.g., employee ID, department, hire date) via REST API.
- **OpenCRX**
  External Customer Relationship Management System.
  Provides sales-related KPIs (e.g., revenue, opportunities, customer interactions) via REST API.

1.3 SmartHoover Performance System

The SmartHoover Performance System acts as an integration backend that:

- Exposes internal performance data via its own REST API (Node.js + Express)
- Stores SmartHoover-specific data in a local MongoDB
- Integrates external data from OrangeHRM and OpenCRX
- Will be used later by a planned Angular frontend

1.4 Context Diagram (Textual Description)

- Users (Salesman, Supervisor, HR Manager) access the SmartHoover Performance System via a browser-based UI (future Angular frontend).
- The SmartHoover backend processes user requests and communicates with two external REST services:
    - OrangeHRM for HR data
    - OpenCRX for CRM/sales KPIs
- The backend also interacts with a local MongoDB where SmartHoover salesmen and performance data are stored.

---

2. Logical / Building Block View

The logical view describes the internal structure of the system and how the components interact.

2.1 Main Components

1. API Layer (api/routes)

- Defines all REST endpoints using Express.js
- Maps HTTP requests to controller functions
- Validates basic request parameters
  Example:
  /api/v1/salesmen, /api/v1/performance, /api/v1/integration/orangehrm

2. Controller Layer (controllers)

- Implements business logic for:
    - Salesman management
    - Performance record management
    - Integration with OrangeHRM & OpenCRX
- Coordinates between API layer, domain models, persistence, and integration modules

3. Domain / Model Layer (models)

- Defines the structure of SalesMan and SocialPerformanceRecord as Mongoose schemas
- Provides validation and type structure
- Ensures consistent data representation

4. Persistence Layer (db)

- Contains MongoDB connection logic (mongo.js)
- Executes CRUD operations using Mongoose models

5. Integration Layer (integration/orangehrm, integration/opencrx)

- REST clients to call:
    - OrangeHRM API (employee HR data)
    - OpenCRX API (sales KPIs)
- Converts external API responses into internal domain structures

6. Security Layer

- Handles authentication (e.g., JWT tokens)
- Authorization for different user roles

7. Configuration / Common Utilities (config, utils)

- Contains environment configuration (Mongo URI, API URLs)
- Logging, error helpers, etc.

2.2 Dependencies (Textual UML Package Description)

- routes → depends on controllers
- controllers → depend on:
    - models
    - integration.orangehrm
    - integration.opencrx
    - db
- models → depend on MongoDB (via Mongoose)
- integration.* → depend on external systems

This structure enables modularity and separation of concerns.

---

3. Process / Runtime View

This view describes how the system behaves at runtime for typical use cases.

3.1 Scenario: “Salesman views his performance overview”

1. The Salesman logs into the UI and selects “My Performance.”
2. UI sends request:
   GET /api/v1/performance/overview?sid={sid}
3. Express.js forwards the request to PerformanceController.getOverview()
4. Controller:
   - Loads performance data from MongoDB
   - Calls OrangeHRM REST API for HR data
   - Calls OpenCRX REST API for sales KPIs
   - Aggregates all data into one JSON response
5. Backend returns the aggregated data to the frontend
6. The UI displays the combined overview to the user

3.2 Scenario: “Supervisor creates a new performance record”

1. Supervisor opens a salesman profile and enters a new performance record
2. UI sends request:
   POST /api/v1/performance
3. Controller validates:
   - Salesman ID
   - Year
   - No existing record for same year & salesman
4. Record is stored in MongoDB through the model
5. Backend returns HTTP 201 Created with the new document
6. UI updates the displayed data

3.3 Scenario: “HR Manager checks HR data inside the system”

1. HR Manager selects “Employee Overview”
2. UI sends:
   GET /api/v1/salesmen/with-hr

3. Backend:
    - Loads local salesman data from MongoDB
    - Calls OrangeHRM API to enrich each entry with HR details
4. Combined dataset is returned to UI

---

4. Deployment View

This view shows how the system is deployed physically.

4.1 Deployment Nodes

1. Client (Browser / Angular Frontend)

- Runs in user's browser
- Sends REST requests to backend

2. SmartHoover Backend (Node.js Server)

- Hosts Express.js API
- Runs business logic, controllers, integration modules
- Communicates with:
    - MongoDB
    - OrangeHRM REST API
    - OpenCRX REST API

3. MongoDB Server

- Stores:
    - Salesmen
    - Social performance records

4. OrangeHRM Server (External)

- Provides employee HR metadata
- Accessed over HTTPS

5. OpenCRX Server (External)

- Provides sales KPIs

- Accessed over HTTPS

4.2 Communication Paths

- Browser → Backend: **HTTP/HTTPS REST**
- Backend → MongoDB: **MongoDB protocol using Mongoose**
- Backend → OrangeHRM: **HTTPS REST**
- Backend → OpenCRX: **HTTPS REST**

4.3 Deployment Diagram (Text Description)

[Browser/Frontend]

↓ HTTP/HTTPS

[Node.js Backend Server]

↓ MongoDB Protocol

[MongoDB Database]

[Node.js Backend Server] ←→ [OrangeHRM REST API]

[Node.js Backend Server] ←→ [OpenCRX REST API]

All external connections should use encryption (TLS/HTTPS).