Project Documentation: Sales Representative Management System

# 1. Architecture Overview

## Technology Stack

- Frontend: Angular 20 (with Standalone Components, Material UI)  
- Backend: ASP.NET Core Web API (.NET 8)  
- Database: SQL Server 2022  
- Reporting: Power BI (Desktop-based dashboard)  
- Hosting-ready: SSR setup   
- Tooling: GitHub for version control, GitHub Desktop for push/pull, Visual Studio + VS Code

## Design Pattern & Structure

Frontend:  
- Modular feature folders (/sales-representatives, /products, /targets, /sales)  
- Angular Material used for responsive UI  
- Reactive Forms used for form control and validation  
- Routing configured with path-based lazy loading  
  
Backend:  
- Layered architecture: Controllers → Services → Repositories  
- DTOs used for input/output mapping  
- Entity Framework Core for DB interaction with Fluent API + Data Annotations  
- Exception Handling Middleware for global error capture  
- CORS enabled for frontend-backend interaction  
  
Database:  
- Normalized schema with primary/foreign keys  
- Referential integrity via FK constraints  
- Indexed commonly filtered columns  
- Date fields follow yyyy-MM-dd format consistency (handled server-side)

# 2. Design Choices & Reasoning

|  |  |
| --- | --- |
| Design Choice | Reasoning |
| Angular Standalone Components | Cleaner, tree-shakable build, modern architecture |
| Reactive Forms | More testable and scalable for validation logic |
| Entity Framework Core | Rapid development with LINQ and migration support |
| DTOs in ASP.NET Core | Prevents over-posting and helps shape API response |
| Power BI Desktop | Easy to integrate, supports drag-and-drop KPIs, good for stakeholder presentations |
| SSR Setup | Improved load speed on initial page visits |

# 3. Testing Methodologies

## Unit Testing

- Framework: xUnit  
- Targets: Controller logic, validation, service layer  
- Mocking: Moq used for mocking dependencies in service/controller tests

## Test Coverage Strategy

- Controllers: Happy path + edge case tests (e.g., invalid IDs, null payloads)  
- Validation: Tests for model validation and BadRequest scenarios  
- Database: Manual test data inserts ,faker data for local test runs

# 4. Deployment Steps

## Local Dev Setup

1. Clone Repo:  
 git clone https://github.com/ganeshkumar2025/flour-assessment-final.git  
2. Backend:  
 - Run dotnet ef database update  
 - Start API via Visual Studio or dotnet run  
3. Frontend:  
 - cd salesrep-manager-ui  
 - npm install  
 - ng serve

## Manual Deployment (if needed)

Backend:  
- Publish via Visual Studio to IIS or Azure App Service  
  
Frontend:  
- Use ng build --configuration production  
- Deploy dist folder to IIS or Azure Static Web Apps

## Power BI

- Open .pbix file in Power BI Desktop  
- Connect to SQL Server DB (adjust credentials if needed)  
- Use “Refresh” to pull latest metrics