Documentation

DEVELOPER

Mrinal Mayank
Department of Computer Science & Engineering
Chandigarh University

DEVELOPED USING SPRING BOOT

SUMMARY

This document provides an overview of the features, setup instructions, and usage guidelines for the Customer Management System built using Spring Boot. The Customer Management System is designed to manage customer data efficiently. It allows users with different roles (ADMIN, MANAGER, and EMPLOYEE) to perform various operations on customer records. The system provides functionalities such as adding, updating, and deleting customer details, as well as pagination and sorting for easy navigation.

Framework Used	Spring Boot 3.1.1
Language	Java 17
Project Management	Maven
Database	MySQL
Frontend	Thymeleaf, HTML, CSS, Javascript

INSTRUCTION TO RUN THE PROJECT:

Requirements

1. MySQL Workbench (for Database)
2. Java Development IDE (IntelliJ Preferred)

Database Structure

There are 2 SQL Scripts inside Database Scripts:

1. **Create-Database-User.SQL:** To create user with password used in **application.properties** to access the database or you can change your username and password in **application.properties** here to connect to database.
   spring.datasource.url=jdbc:mysql://localhost:3306/customer_management
   spring.datasource.username=**customeradmin**
   spring.datasource.password=**Test@123**

2. **Create-Customer-Management-Tables.SQL:** It is **mandatory** to run this SQL Script to create all database tables used in the project inside connection authenticated with above credentials.

   There are 3 tables with sample data:
   a. **Customer Table:** Customer Details like id, first_name, last_name etc.
   b. **Users Table:** Store email & password to authenticate user to theManagement System.
   c. **Roles Table:** Mapping Roles with email like EMPLOYEE, MANAGER & ADMIN to grant them access to specific operations in the application.

Project Structure

Inside **main.java.com.sunbasedata.customerhub** there are 5 Packages:

1. **Model Layer (Entity):** Customer Model with Constructors & Getter Setters
2. **Service Layer (Service):** Services Operations Like find, findById, GetAllCustomers etc.
3. **Configuration Layer (Security):** Specifying from which table to fetch authentication credentials and mapping endpoints with roles
4. **Rest (Controller Layer):** Controller to call service and return result in the frontend.
5. **JPA Repository (DAO):** Contains Data Access Object (DAO) classes that interact with the database using JPA (Java Persistence API).

Inside main.java there is another folder **resources.templates** with Frontend Templates

Now you just have to make the **database configuration** to successfully run the project inside **application.properties**, make sure to run **Create-Database-User.SQL** to create databases and provide appropriate datasource credentials in properties.

Roles Mapping

Functions	GET All Customers	Get Customer by ID	Pagination & Sorting	Add Customer	Update Customer	Delete Customer
ADMIN	Authorized	Authorized	Authorized	Authorized	Authorized	Authorized
MANAGER	Authorized	Authorized	Authorized	Authorized	Authorized	Unauthorized
EMPLOYEE	Authorized	Authorized	Authorized	Unauthorized	Unauthorized	Unauthorized

Don't worry to provide roles, **Create-Database-User.SQL** already created username and password and mapped desired roles. Authentication credentials to access all operations in the system (ADMIN):

- **Email: test@sunbasedata.com**
- **Password: Test@123**