**Case Study Problem**

**Name:** Lakshya Joshi

**CT/DT Reference No:** dt20173831354

**College:** Chandigarh Engineering College, Landran

**Email ID:** lakshya.joshi002@gmail.com

**Phone No:** 8146050920

**Provide the Problem Statement that you have chosen:**

**Employee Management tool**

# Dodo want to start a company and he had hired many people as per required skill so now he is finding it difficult to manage associates, so he needs a small help from you.

# Design a low level design that cover all the scenarios to add an employee in DB and update the record if required and delete the record if associate is no longer part of the dodo's little world.

# The following is list of help that Dodo needs from you:

# Design a low level design covering all the scenarios he need on the db.

# Add employee in the employee table and the designated table.

# If associate got promoted then the designation should be updated in the required table

# There should not be any ambiguity in the data

# Only authorized person can delete the record from the table[If any associate is no longer part of dodo's world then only manager and Dodo himself can delete that record]

# Deletion should update the associate in ex-associate table

# Create Junit for each class to have perfect flow of services

**List of micro services need to develop using above Tools, Language and Frameworks**

GET - List of employee

POST - Add a new Employee

PUT- Update record of employee

DELETE- delete an employee

**Approach to the solution:**

First of all thank you for giving me such an exciting opportunity, I feel privileged to perform this task. It helped me in exploring many horizons of the technology, which I wasn’t aware about. This task was challenging to me because it had a time constraint of 1 week and I just had two days(Saturday and Sunday) to work on it. Still I have tried to give my best on this task.

Below is the approach I followed to complete this task.

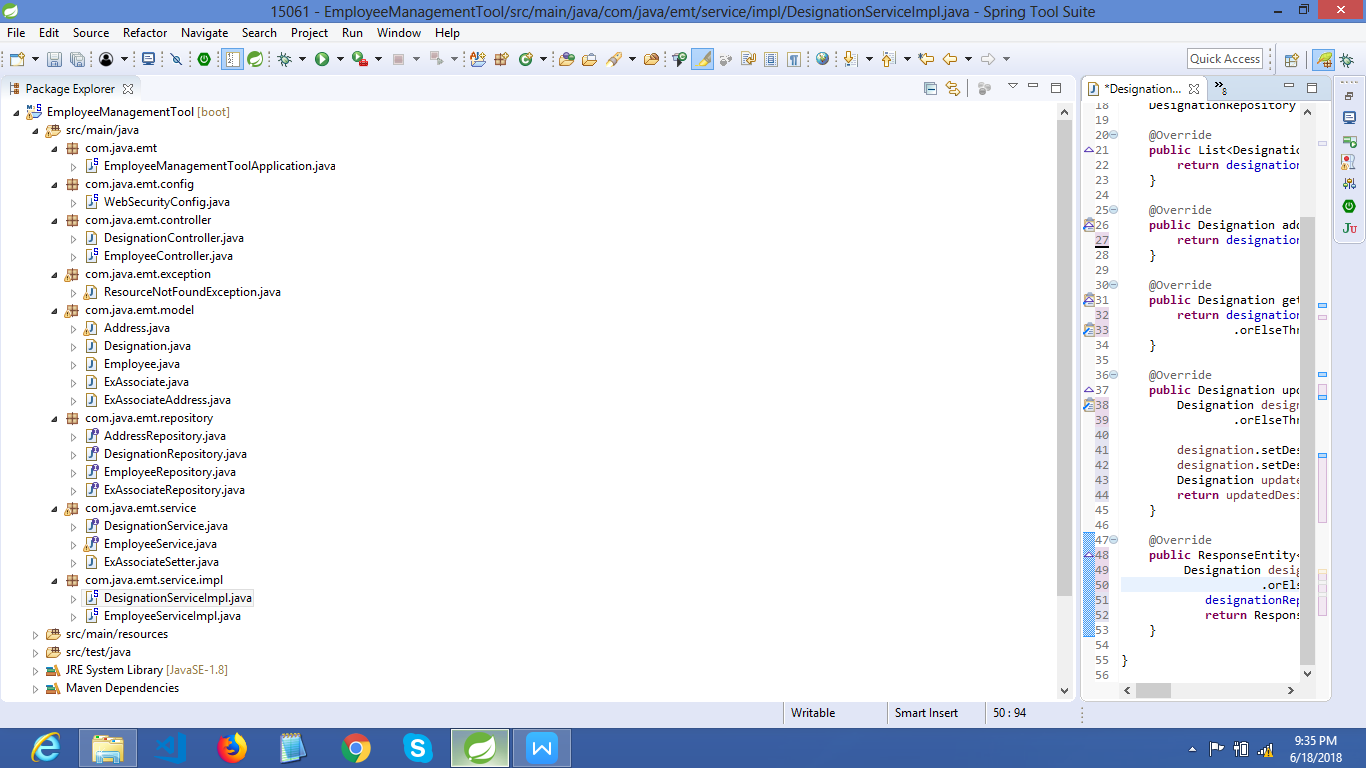
To fulfil DODO’s needs I have tried to maintain the modularity of the code, with strong emphasis given on the naming conventions to make my code look clean and readable I have attached my project structure in the next section too. Also I tried to give him an optimal solution with no redundant data in the database. There is an adequate amount of normalization performed in the database to keep DB clean and this is done using Jpa/Hibernate. I have prepared an ER diagram of my database structure which I will attach in the next section.

Below are specific approaches to the given requirements:

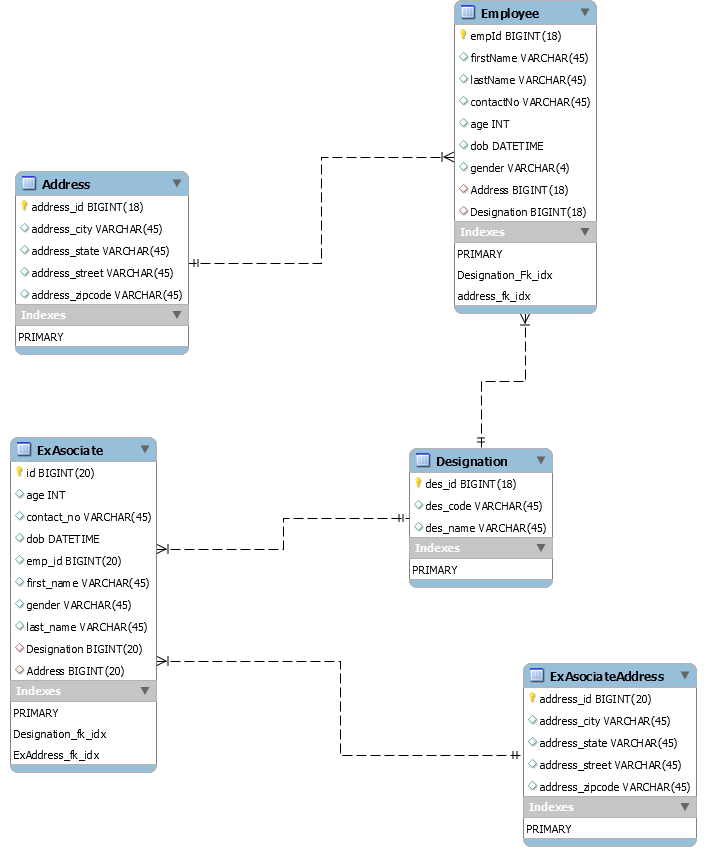
1. DB interactions are performed using “JpaRepository”.
2. Employee is added in the employee table, with a foreign key referring to an address table which is mapped “One to One” with employee table. Also there is “Many to One” mapping of employees with designation table, which consist of information regarding all the designations in the DODO’s company. This approach reduced a lot of redundancy from database as there are no multiple entries of different employees with same designation.
3. To perform authorization before deleting any record of employee from database, I have used Spring Security’s inMemoryAuthorization which only allows “Admin” that is DODO and “Manager” to perform deletion from database.
4. When employee gets deleted from the database, It gets updated in the Ex Associate table.
5. To run this project you need to provide credentials of your database in application.properties file. Also before starting to add employees, designations should be added in the Designation table.

**Attach the screenshot of the design/solution:**

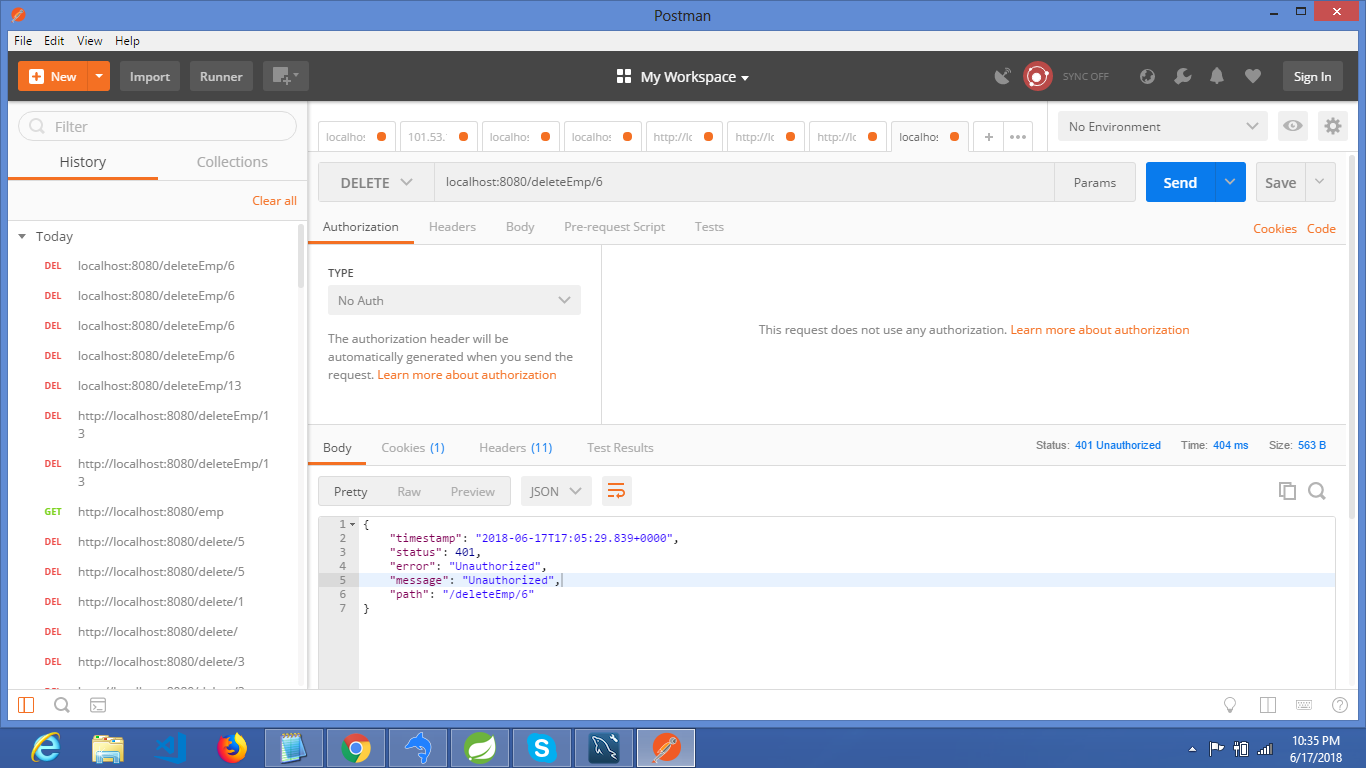
**PROJECT STRUCTURE**



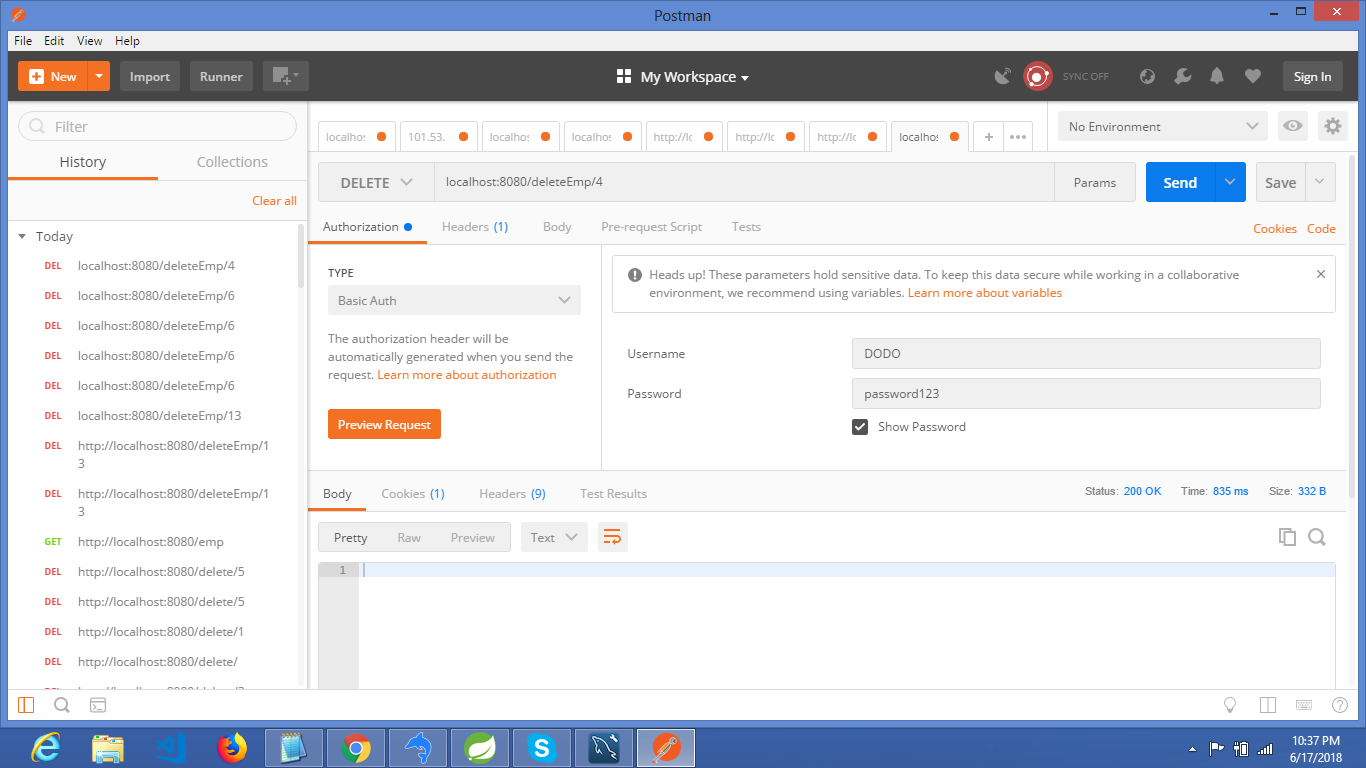
**Employee Management ER Diagram**

****

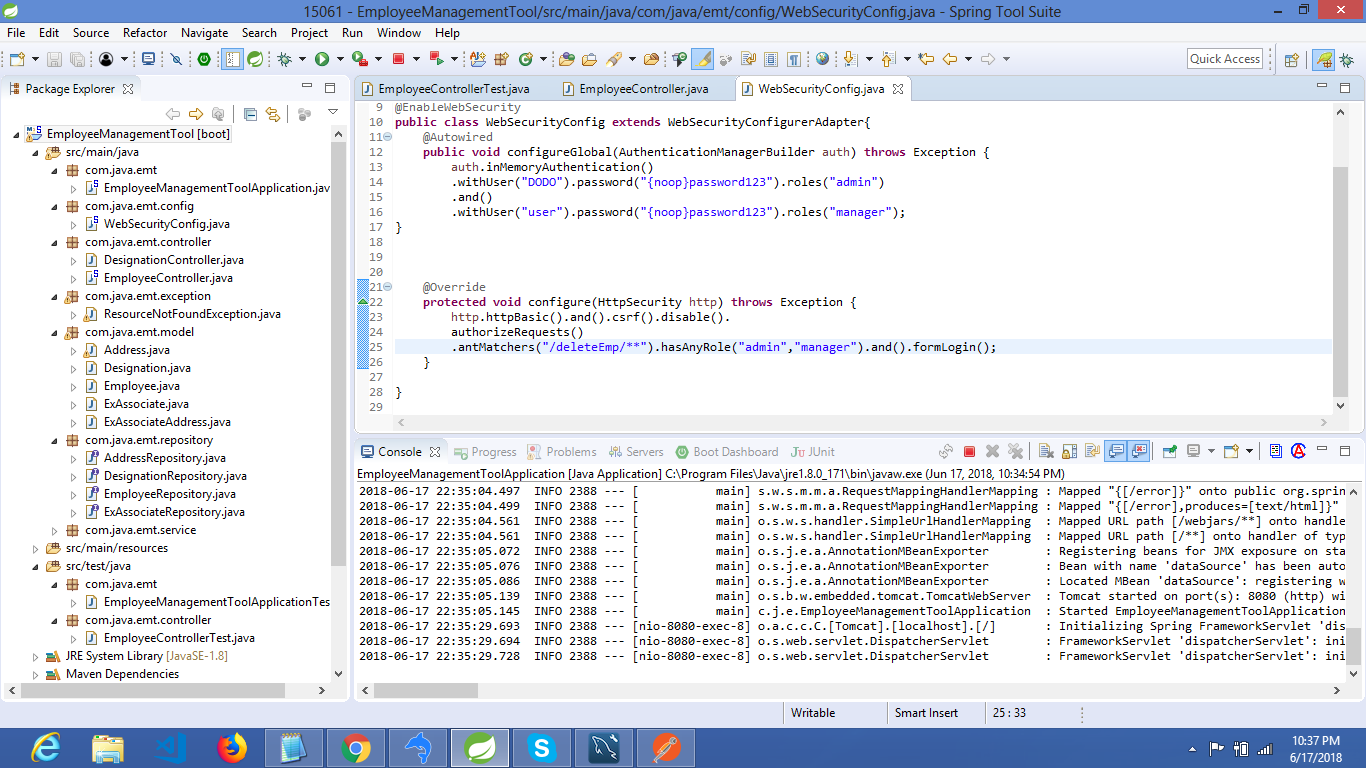
**Unauthorized user can’t access delete capability (Status 401)**

****

**Authorized DODO can access delete (Status 200 ok)**

****

**Spring security**

****