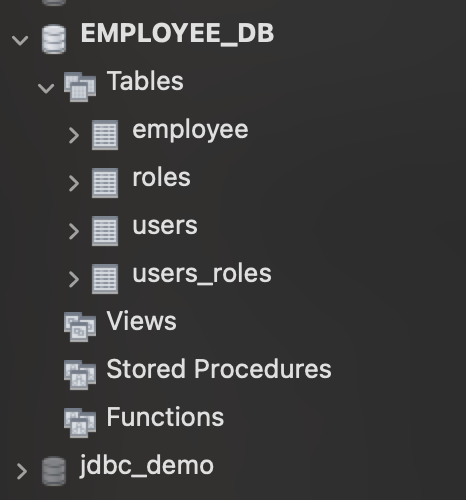
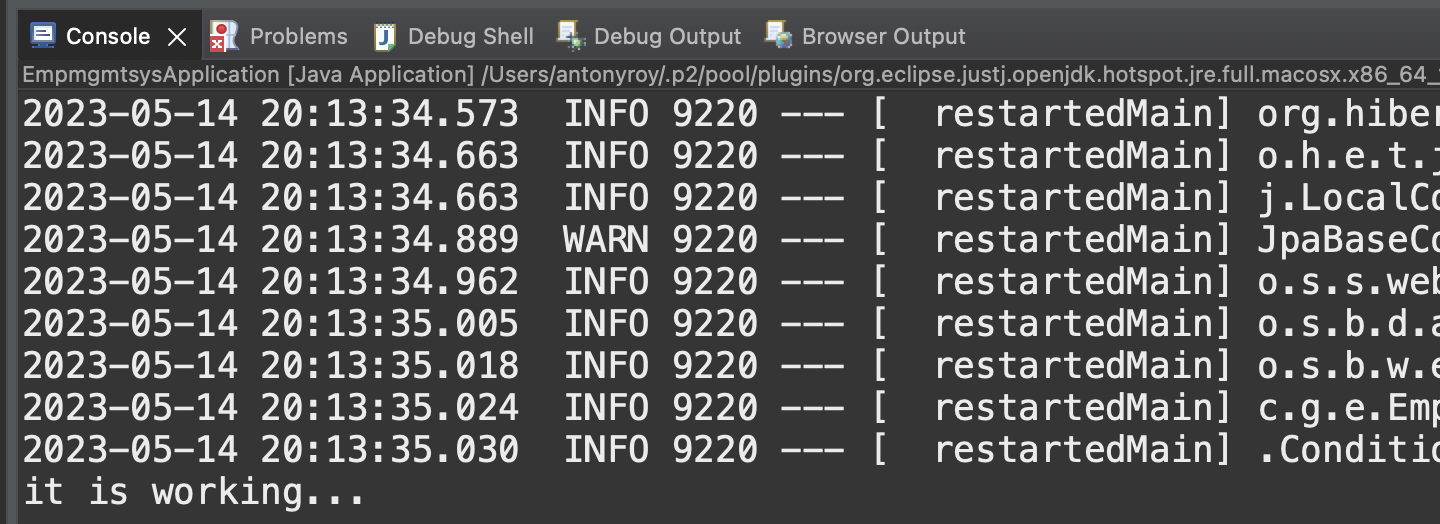
**Group Assignment 4 - Testing**

Execute the db scripts to create a database



Create database employee\_db;

**Run the application:**

****

**Result :SUCCESSFUL**

Open the Postman to execute the following steps:

|  |  |
| --- | --- |
| Test No#1 | Your application should be able to add roles in the database dynamically in the db.  --1. Add ‘ADMIN’ ROLES. (POST)  -- url : localhost:8080/api/role/  **POSTMAN**  {"name":"ADMIN"}    {"name":"USER"}    **DATABASE** |
| Result: | SUCCESSFUL |

|  |  |
| --- | --- |
| Test No#2 | Your application should be able to add Users in the db which can be used for authentication purposes.  --1. Add two usrs - ‘kumar’ & ‘amit’. (POST)  -- url : localhost:8080/api/user  Add to users  {"username":"kumar","password":"123"}    {"username":"amit","password":"123"}    Database: |
| Result: | SUCCESSFUL |

|  |  |
| --- | --- |
| Test No#3 | 3. Now Your application should be able to add employees data in the db if and only if the authenticated user is **ADMIN**-  -- end point :localhost:8080/api/employees/  {"firstName":" Aditya ","lastName":" gupta","email":"adityagupta@gmail.com"}  {"firstName":"Bhaskar","lastName":" Raj","email":"BhaskarRaj@gmail.com"}  {"firstName":"Chandra","lastName":"Sekhar","email":"Chandrasekhar@gmail.com"}  {"firstName":"Patrick","lastName":"John","email":"Patrickjohn@gmail.com"}  {"firstName":"Smitha","lastName":"Mohan","email":"Smithamohan@gmail.com"} |
| Result: | Postman Result:            5 records added  Database Result:    Result : Successful |

|  |  |
| --- | --- |
| Test No#4 | 4. Your application should provide an endpoint to list all the employees stored in the database.  url : localhost:8080/api/employees/ (GET) |
| Result: |  |

|  |  |
| --- | --- |
| Test No#5 | 5. Your application should provide endpoint to fetch or get an employee record specifically based on the id of that employee-  url : localhost:8080/api/employees/1 (GET) |
| Result: | Postman :  Retrieving the 1st record    Result : successful |

|  |  |
| --- | --- |
| Test No#6 | 6. Your application should provide an endpoint to update an existing employee record with the given updated json object.  url : localhost:8080/api/employees/1 (PUT)  update the 1st record to Arun Kumar  {  "id": 1,  "firstName": "Arun",  "lastName": "Kumar",  "email": "ak@gmail.com"  }    Database : |
| Result: | Successful |

|  |  |
| --- | --- |
| Test No#7 | 7. Your application should also provide an endpoint to delete an existing employee record based on the id of the employee-  Delete an employee no.2  url : localhost:8080/api/employees/2 (DELETE) |
| Result: | Postman:    Database:  Record 2 is deleted from the database |

|  |  |
| --- | --- |
| Test No#8 | 8. Your application should provide an endpoint to fetch an employee by his/her first name and if found more than one record then list them all-  url : localhost:8080/api/employees/search/ Chandra  Postman |
| Result: | Successful |

|  |  |
| --- | --- |
| Test No#9 | 9. Your application should be able to list all employee records sorted on their first name in either ascending order or descending order .  (a) Url- [http://localhost:8080/api/employees/sort?order=”asc](http://localhost:8080/api/employees/sort?order=%E2%80%9Dasc)”  OR  (b)Url- http://localhost:8080/api/employees/sort?order=”desc” |
| Result: | Postman   1. Sorted firstname in Ascending order      1. Descending Order     Result : Successful |