|  |  |  |
| --- | --- | --- |
| **S.No.** | **Scenario** | **Actual result** |
| 1. | Your application should be able to add roles in the database dynamically in the db.  Ex-  {  "name":"USER"  }  Where name specifies a role which can be assigned to a user that will be used for authentication purposes while interacting with the api. |  |
| 2 | Your application should be able to add Users in the db which can be used for authentication purposes.  Ex-  {  "username":"temp",  "password":"12345",  "roles":[{  "id":2,  "name":"USER"  }]  } |  |
| 3 | Now Your application should be able to add employees data in the db if and only if the authenticated user is **ADMIN**-  Ex-  {  "firstName":"gl",  "lastName":"postman",  "email":"postman@gamil.com"  } | With user/password as “admin/admin”      With user/password as “user1/password01” |
| 4 | Your application should provide an endpoint to list all the employees stored in the database. |  |
| 5 | Your application should provide endpoint to fetch or get an employee record specifically based on the id of that employee-  Ex- Url- <http://localhost:8080/api/employees/3> |  |
| 6 | Your application should provide an endpoint to update an existing employee record with the given updated json object.  Object to be updated(raw>Json)-  {  "id":1,  "firstName":"postman",  "lastName":"postman",  "email":"postman@gamil.com"  }  Response Body after updation-  {  "id": 1,  "firstName": "postman",  "lastName": "postman",  "email": "postman@gamil.com"  } |  |
| 7 | Your application should also provide an endpoint to delete an existing employee record based on the id of the employee-  Ex-  Url- <http://localhost:8080/api/employees/4>  Response Body-  "Deleted employee id - 4” |  |
| 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-  Ex- Url- http://localhost:8080/api/employees/search/gl  Response Body-  [  {  "id": 11,  "firstName": "gl",  "lastName": "postman",  "email": "postman@gamil.com"  }  ] |  |
| 9 | Your application should be able to list all employee records sorted on their first name in either ascending order or descending order .  Ex-  Url- [http://localhost:8080/api/employees/sort?order=”asc](http://localhost:8080/api/employees/sort?order=%E2%80%9Dasc)”  OR  Url- http://localhost:8080/api/employees/sort?order=”desc” |  |
| 11 | You should use the H2 In Memory database/MySQL for the whole project along with Spring JPA and Spring Security. |  |
| 12 | Spring Boot Application must follow the standard project structure . |  |