Table Of Contents

1. Naudit Project………………………………………………………………2

1.1 Project Setup………………………………………………………2

2. Employee List…………………………………………………………….….2

2.1 Project Codes of Employee List…………………………….2

3.Employee List Task…………………………………………………………3

3.1. Creating a Table with our own fields……………………3

3.2. Connection to the Database………………………………..4

3.3. Connecting UI to the backend via Postman………….5

3.3.1 What is Spring Boot…………………………………5

3.3.2 What are Crud operations……………………….5

3.4. Standard CRUD Operation……………………………………6

4.HrStatus Task………………………………………………………………….7

4.1. Creating a Table with our own fields……………………7

4.2. Connection to the Database…………………………….….8

4.3. Connection to the UI to backend via postman.…..9

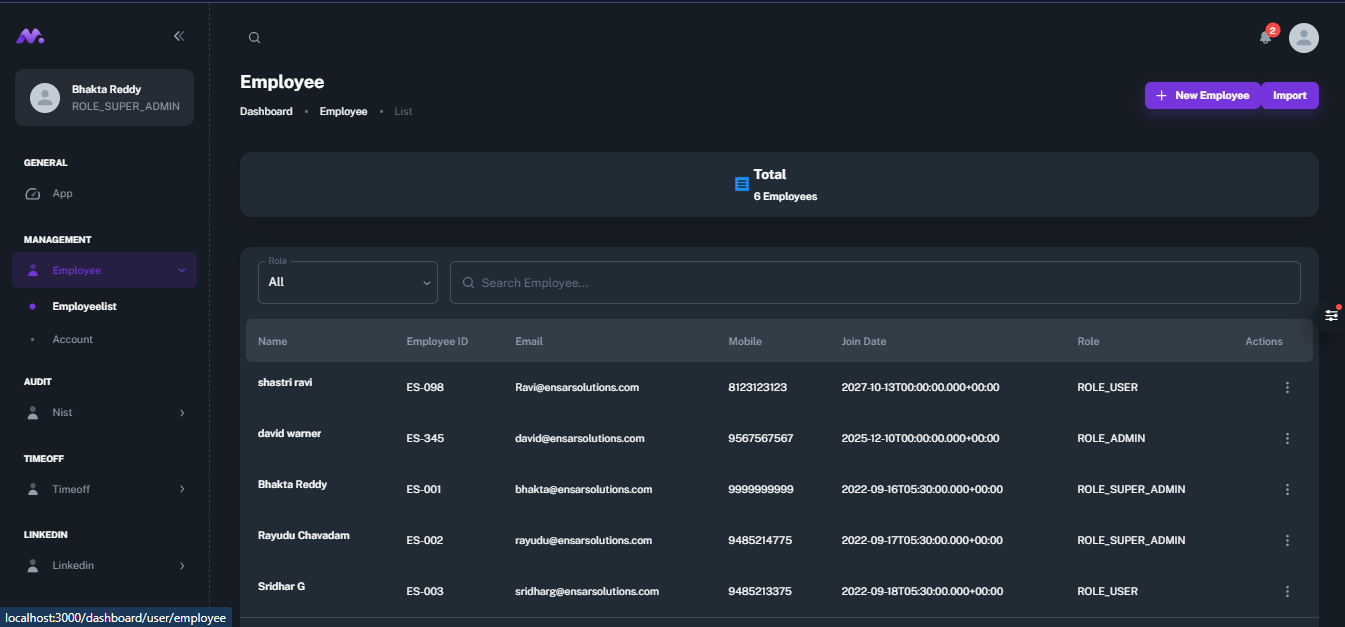
1.Naudit Project:

**1.1 Project Setup:**

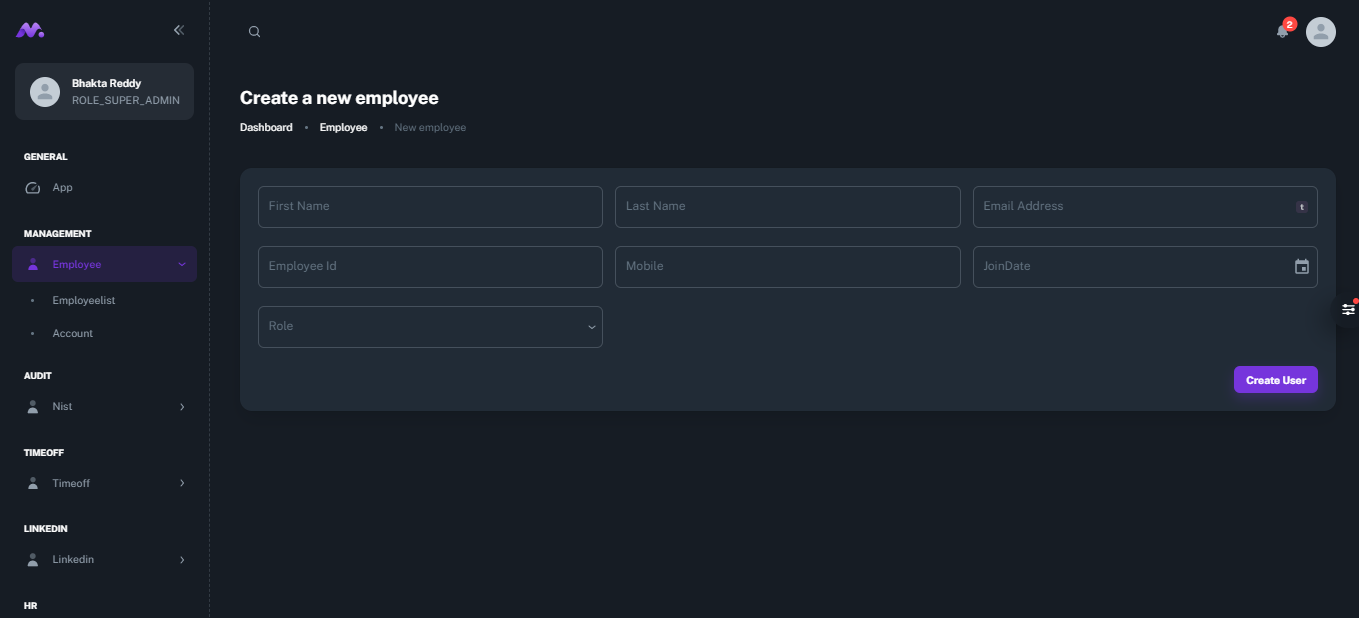
* First, I cloned NauditUI and NauditAPI from Nproducts organization.
* I connected NauditUI in Visual Studio Code and installed Node Modules in it.
* Secondly, I opened NauditAPI in Eclipse and run it as Gradle tasks.
* Later, I connected SQL Workbench by giving SQL Password.

2. Employee List:

* In Employee list, you can do crud operations by creating, reading, updating and deleting data.
* This is the Home Page of the Employee page. You can see the Data here.



* This is the Create Page.



3. Employee List task:

3.1 Creating a Table-

**CREATE** **TABLE** `user`

(

`id` char(36) **NOT** NULL,

`first\_name` varchar(50) NULL,

`last\_name` varchar(50) **NOT** NULL,

`employee\_id` varchar(50) **NOT** NULL,

`mobile` char(12) **NOT** NULL,

`join\_date` timestamp **NOT** NULL **DEFAULT** CURRENT\_TIMESTAMP,

`email` varchar(50) **NOT** NULL,

`password` varchar(500),

`role\_name` varchar(500) **NOT** NULL **DEFAULT** 'ROLE\_USER',

`role\_id` char(36) null **references** role(`id`),

`disabled` boolean **NOT** NULL **default** **false**,

`created\_date\_time` timestamp **NOT** NULL **DEFAULT** CURRENT\_TIMESTAMP,

`last\_updated\_date\_time` timestamp NULL **DEFAULT** NULL,

`organization\_id` char(36) **not** null **references** organization(`id`),

**PRIMARY** **KEY** (`id`),

**UNIQUE** **KEY** `user\_email\_unique` (`email`)

) ENGINE = InnoDB

**DEFAULT** CHARSET = utf8;

**DROP** **TABLE** **IF** **EXISTS** `user\_password\_reset\_request`;

3.2 Connection to Database-

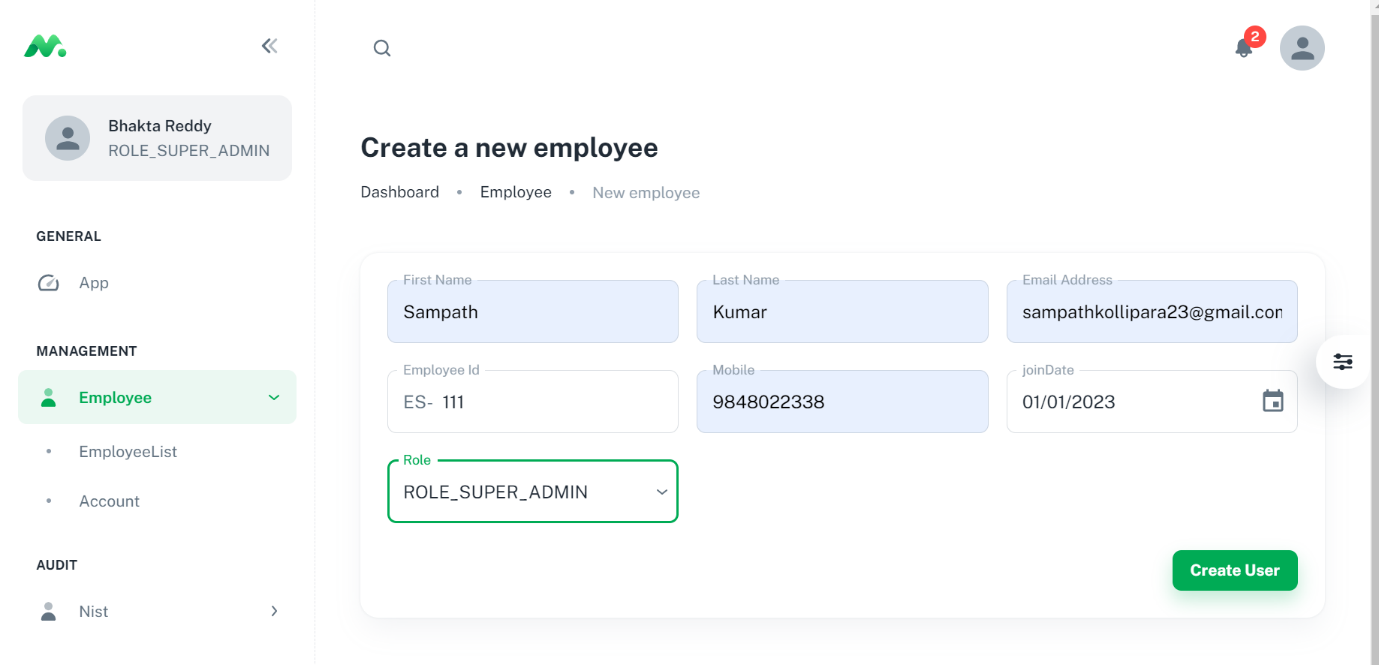
* + I connected MySQL Workbench to Eclipse by giving the MySQL password in the Application Properties Section.
  + Here, we can change the port address as our wish.
* JWT\_SECRET\_KEY=abhra\_ka\_dhabra
* JWT\_TTL\_MINS=180
* DB\_SERVER=localhost
* DB\_PORT=3306
* DB\_SCHEMA=audit
* DB\_USE\_SSL=false
* DB\_REQUIRE\_SSL=false
* DB\_USER=root
* DB\_PASSWORD=Dimpu123@
* EMAIL\_FROM=<from\_email>
* EMAIL\_USER\_NAME=<from\_user\_name>
* EMAIL\_USER\_PWD=<email\_pwd>
* EMAIL\_HOST=<email\_Host>
* EMAIL\_PORT=<emap\_port>
* AWS\_ACCESS\_KEY=<access\_key>
* AWS\_SECRET\_KEY=<access\_secret>
* AWS\_ACCOUNT\_ID=<aws\_act>
* AWS\_USER\_ARN=<arn\_url>
* spring.redis.host=localhost
* spring.redis.port=6379
* spring.redis.database=0
* kafka.bootstrapAddress=localhost:29092

3.3 Connecting UI to Backend Via Postman:

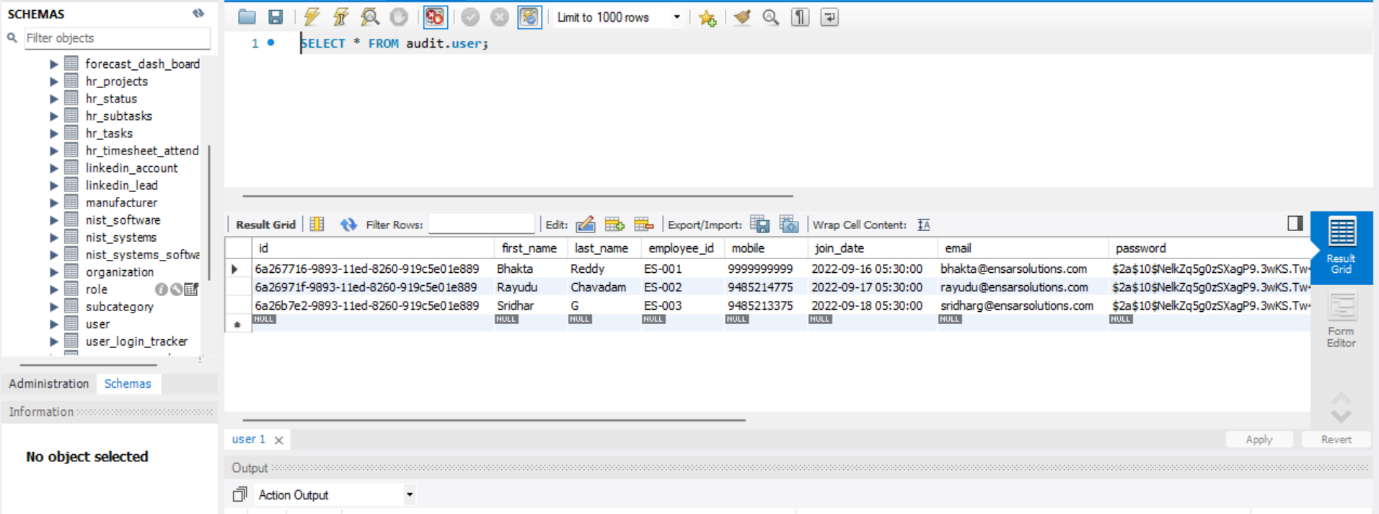
* + Here, I am using Postman to connect Frontend and Backend by using Get, Post and Put methods like this:
    - GET: <http://localhost:8080/v1/user/>
    - POST: http://localhost:8080/v1/user/
    - PUT: http://localhost:8080/v1/user/
    1. What is Spring Boot?
* Spring boot is a module of spring framework which is used to create stand-alone, production-grade Spring based Applications with minimum programmer’s efforts. It is developed on top of core spring framework. The main concept behind spring boot is to avoid lot of boilerplate code and configuration to improve development, unit test etc.
  + 1. What are Crud operations?
* The **CRUD** stands for **Create, Read/Retrieve, Update,** and **Delete**. These are the four basic functions of the persistence storage.
* The CRUD operation can be defined as user interface conventions that allow view, search, and modify information through computer-based forms and reports. CRUD is data-oriented and the standardized use of **HTTP action verbs**. HTTP has a few important verbs.
* The Crud Operations are:
* **POST:** Creates a new resource
* **GET:** Reads a resource
* **PUT:** Updates an existing resource
* **DELETE:** Deletes a resource
  1. employee Id Prefix:
* Here, I added Prefix of ES- to employee.java in Spring Boot.

user.setEmployeeId("ES-" +createUpdateUserDto.getEmployeeId());

* **Before Creation of Data:**



**Stored in Database:**



4.HRStatus Task:

4.1. Creating a Table with Our own Fields:

**DROP** **TABLE** **IF** **EXISTS** `hr\_status`;

**CREATE** **TABLE** `hr\_status`

(

`id` char (20) **NOT** NULL,

`name` varchar(40) **NOT** NULL,

`title` varchar(50) **NOT** NULL,

`task` varchar(50) **NOT** NULL,

`tags` varchar(50) **NOT** NULL,

`start\_time` varchar(50) **NOT** NULL,

`end\_time` varchar(50) **NOT** NULL,

`attachments` varchar(40) **NOT** NULL,

`work\_date` DATE NULL,

`description` varchar(2000) NULL,

`user\_id` char(36) null **references** user(`id`),

`projects\_id` char(36) null **references** hr\_projects(`id`),

`created\_date\_time` timestamp **NOT** NULL **DEFAULT** CURRENT\_TIMESTAMP,

`last\_updated\_date\_time` timestamp NULL **DEFAULT** NULL,

**PRIMARY** **KEY** (`id`)

) ENGINE = InnoDB

**DEFAULT** CHARSET = utf8;

4.2 Connection to the Database:

JWT\_SECRET\_KEY=abhra\_ka\_dhabra

JWT\_TTL\_MINS=180

DB\_SERVER=localhost

DB\_PORT=3306

DB\_SCHEMA=audit

DB\_USE\_SSL=false

DB\_REQUIRE\_SSL=false

DB\_USER=root

DB\_PASSWORD=Dimpu123@

EMAIL\_FROM=<from\_email>

EMAIL\_USER\_NAME=<from\_user\_name>

EMAIL\_USER\_PWD=<email\_pwd>

EMAIL\_HOST=<email\_Host>

EMAIL\_PORT=<emap\_port>

AWS\_ACCESS\_KEY=<access\_key>

AWS\_SECRET\_KEY=<access\_secret>

AWS\_ACCOUNT\_ID=<aws\_act>

AWS\_USER\_ARN=<arn\_url>

spring.redis.host=localhost

spring.redis.port=6379

spring.redis.database=0

kafka.bootstrapAddress=localhost:29092

4.3. Connection to the Ui Via Postman:

* Here, I am using postman to send API’s and to connect both UI and Backends like this:
* GET: http://localhost:8080/v1/hrstatus/
* POST: http://localhost:8080/v1/hrstatus/
* PUT: <http://localhost:8080/v1/hrstatus/>
* HRstatus Home page and Data Creation Page:

