







Book Management System

Technology Stack: Spring Boot + JWT

Project Description

A secure RESTful API built using **Spring Boot** that allows authenticated users to manage a collection of books. It supports **JWT-based authentication**, basic **CRUD operations**, and is documented using **Swagger**.

Features

-  **User Registration and Login** with JWT authentication
 -  **Secure Endpoints** using JWT Token
 -  **Book Management Operations:**
 - Add a new book
 - View all books
 - View a book by ISBN
 - Update a book by ISBN
 - Delete a book by ISBN
 -  **Input Validation** and **Global Exception Handling**
 -  **Unit Testing** for the service layer
 -  **API Documentation** using Swagger UI
-

Tech Stack

- Java 17
 - Spring Boot 3
 - Spring Security
 - JWT (JSON Web Token)
 - Spring Data JPA
 - Hibernate
 - MySQL Database
 - Swagger (Springdoc OpenAPI)
-

Project Structure

bash

CopyEdit

com.example.demo

- ├── config # Security & JWT configuration
- ├── controller # REST endpoints
- ├── model # Book entity
- ├── repository # JPA repository interface
- ├── service # Business logic layer
- ├── exception # Custom and global exceptions
- └── dto # Request and Response DTOs

Authentication Flow

1. Register or login using:
 - /auth/register
 - /auth/login
2. Receive a **JWT token** upon successful login.
3. Include the token in the Authorization header for subsequent requests:

makefile

CopyEdit

Authorization: Bearer <your-token>

Example API Usage

Login

http

CopyEdit

POST /auth/login

Content-Type: application/json

{

```
"email": "user@example.com",  
"password": "password123"  
}
```

Add Book

http

CopyEdit

POST /addbook

Authorization: Bearer <your-jwt-token>

Content-Type: application/json

```
{  
  "title": "Effective Java",  
  "author": "Joshua Bloch",  
  "isbn": 123456,  
  "publicationYear": 2008  
}
```

Swagger UI

Access Swagger documentation at:

<http://localhost:8080/swagger-ui/index.html>

How to Run the Project

1. Clone the Repository

bash

CopyEdit

```
git clone https://github.com/your-username/book-management-system.git
```

```
cd book-management-system
```

2. Build and Run the App


bash

CopyEdit

```
mvn spring-boot:run
```

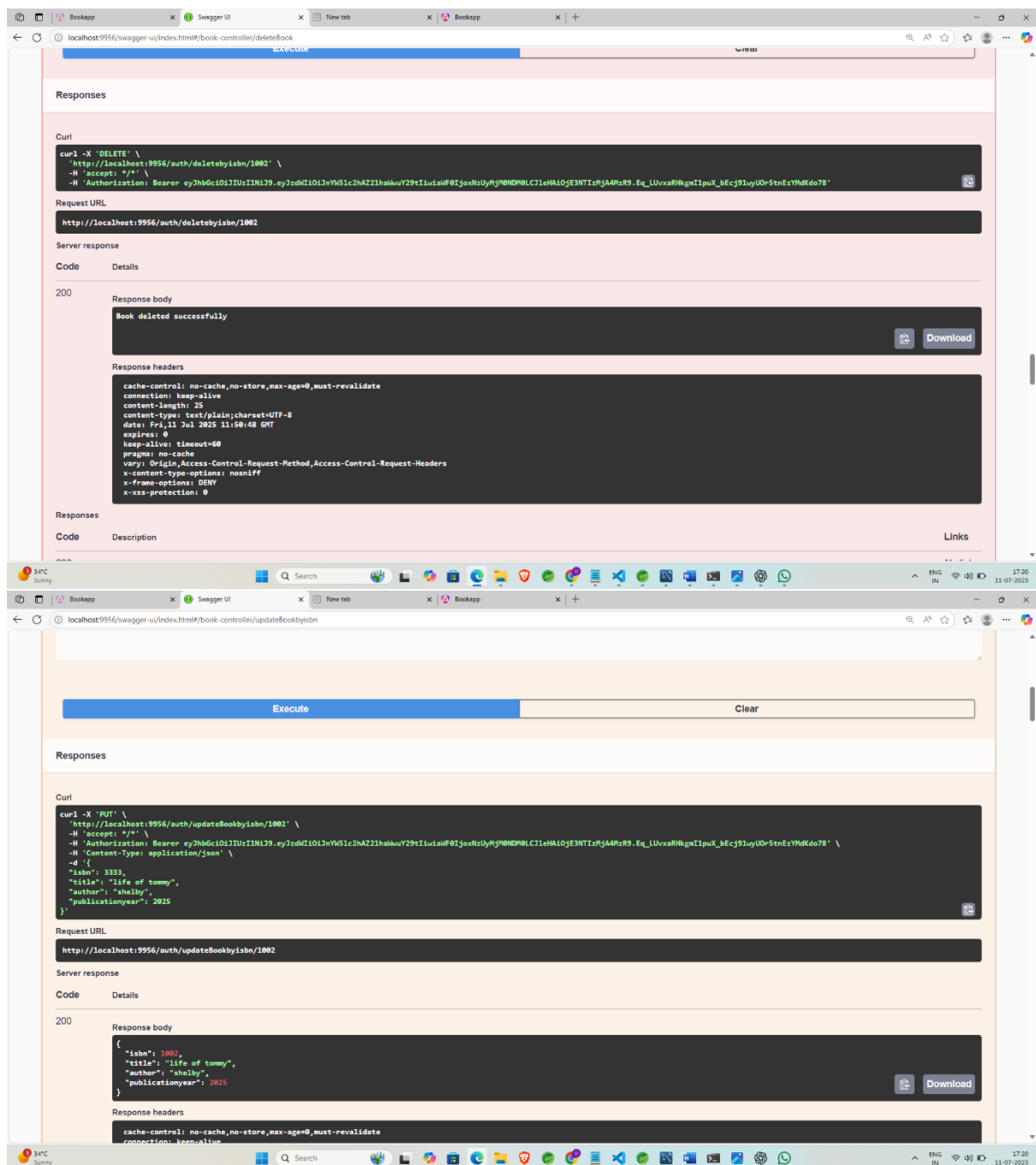
3. Test the API using Swagger UI or Postman.

Author

- **Udhaya Moorthy**
-  Skills: Spring Boot | Java | REST APIs | JWT Security

PPT Link:

[Google Slides - Book Management SystemS](#)



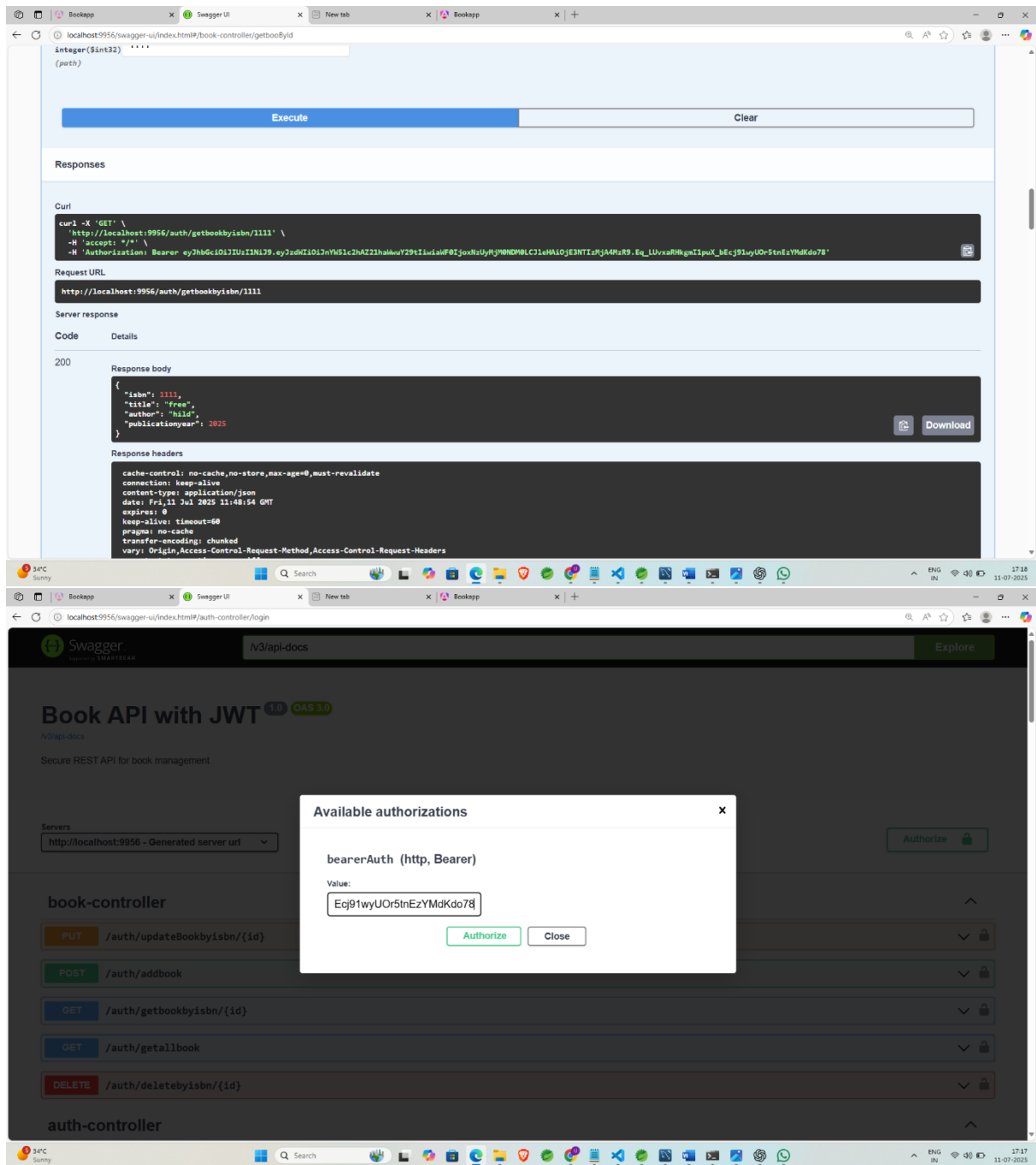
The image displays two screenshots of the Swagger UI interface, showing the results of API tests performed using the 'Execute' button.

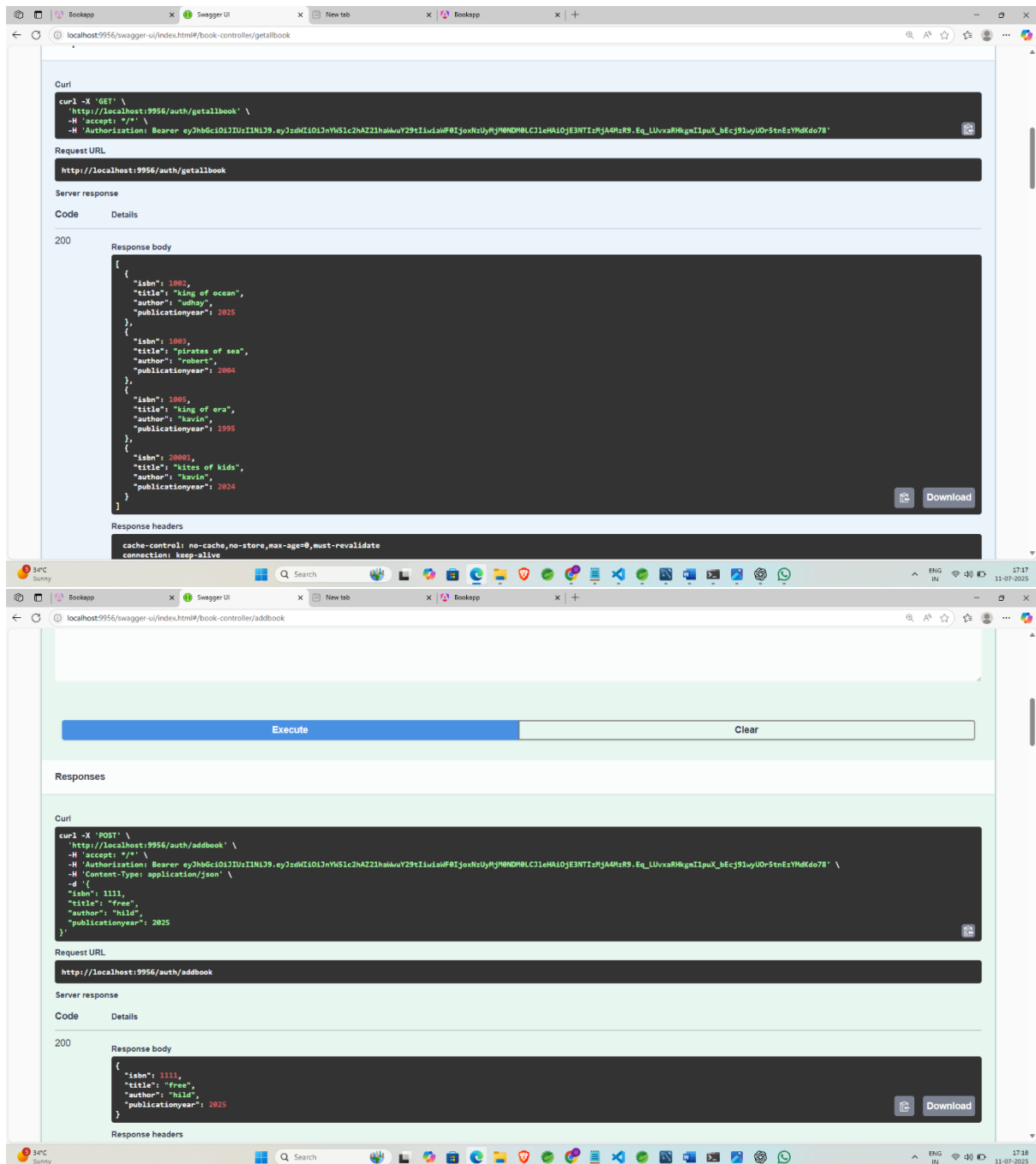
Top Screenshot (DELETE Method):

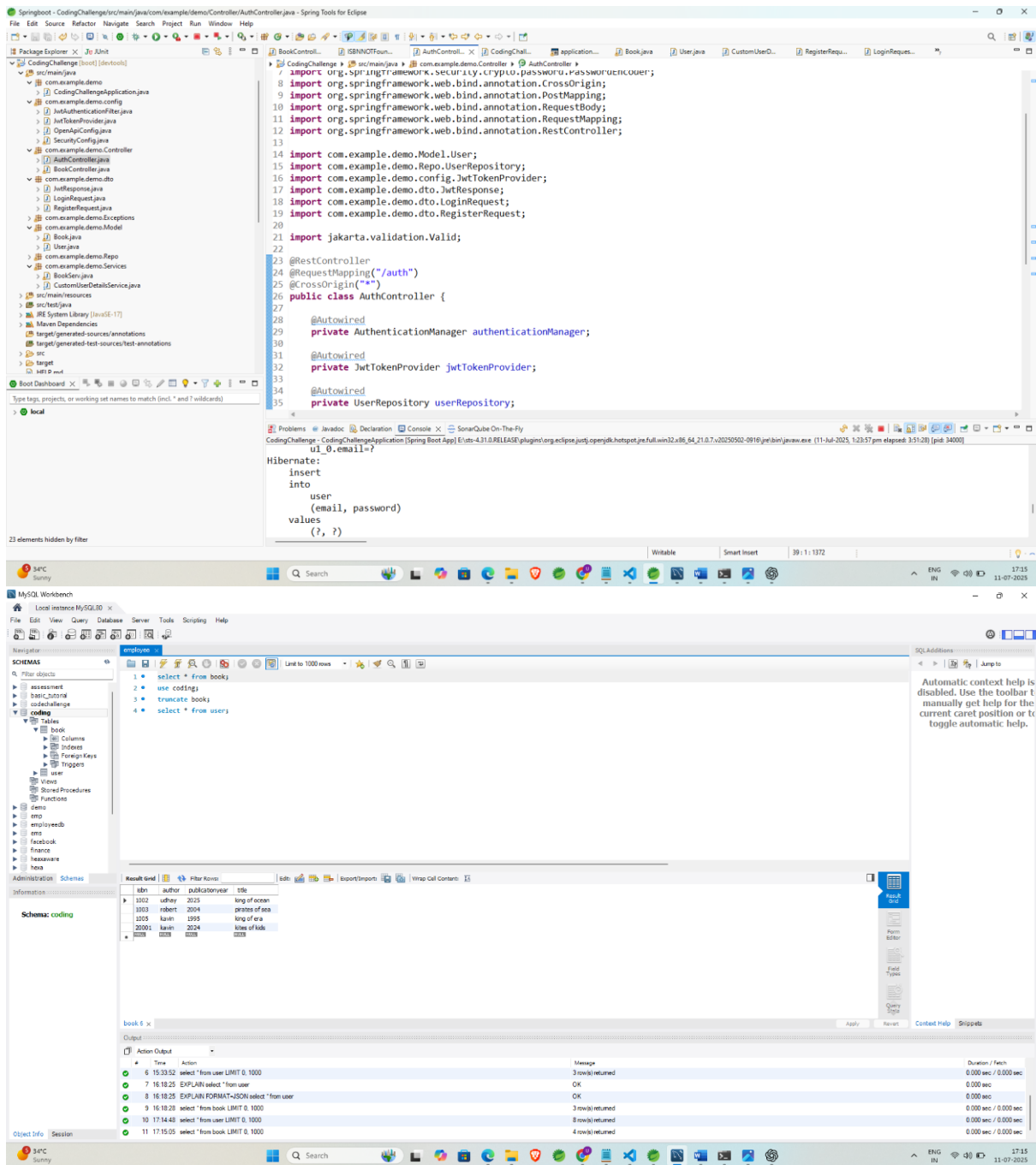
- Request URL:** `http://localhost:9956/auth/deletebyisbn/1002`
- Response Code:** 200
- Response Body:** `Book deleted successfully`
- Response Headers:**
 - `cache-control: no-cache, no-store, max-age=0, must-revalidate`
 - `connection: keep-alive`
 - `content-length: 25`
 - `content-type: text/plain; charset=UTF-8`
 - `date: Fri, 11 Jul 2025 11:58:48 GMT`
 - `expires: 0`
 - `keep-alive: timeout=60`
 - `pragma: no-cache`
 - `vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers`
 - `x-content-type-options: nosniff`
 - `x-frame-options: DENY`
 - `x-xss-protection: 0`

Bottom Screenshot (PUT Method):

- Request URL:** `http://localhost:9956/auth/updateBookbyisbn/1002`
- Response Code:** 200
- Response Body:** `{ "isbn": 1002, "title": "life of tommy", "author": "shelby", "publicationyear": 2025 }`
- Response Headers:**
 - `cache-control: no-cache, no-store, max-age=0, must-revalidate`
 - `connection: keep-alive`







MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

Filter objects

- assessment
- basic_training
- codechallenge
- coding
 - book
 - columns
 - indexes
 - foreign_keys
 - triggers
 - user
 - views
 - stored_procedures
 - functions
- demo
- emp
- employeesdb
- emp
- facebook
- finance
- hessware
- hena

Administration Schemas

Schema: coding

Result grid

id	email	password	role
1	udhaya@gmail.com	\$2a\$10\$gHwzVnH5A9CH-fcCtUgH-ApIq...	User
2	shing	\$2a\$10\$A4nQyCWAUDV72H6J3ODv55KA...	...
3	user11@gmail.com	\$2a\$10\$PmPgCaAQ6959fGwRbWJfY...	...
4	udhayakv@gmail.com	\$2a\$10\$Qv52762vwaqLQ5A8dufyG39f...	...
5	user21@gmail.com	\$2a\$10\$MNTUyQ3X3DtbP938LFR8-q43...	...
6	udhayamathy2018@gmail.com	\$2a\$10\$W4fYwUj2J8mvd9p3uRKA-Fggs...	...
7	user33@gmail.com	\$2a\$10\$4c3C2E94LRm7yB-V5SOSF58F...	...
8	udhayakv12003@gmail.com	\$2a\$10\$G4MP8H9CounQJAApQmRgRNGC...	...

user 5

Output

Time	Action	Message	Duration / Fech
5 14:17:39	select * from book LIMIT 0, 1000	0 rows returned	0.000 sec / 0.000 sec
6 15:23:52	select * from user LIMIT 0, 1000	3 rows returned	0.000 sec / 0.000 sec
7 15:10:25	EXPLAIN select * from user	OK	0.000 sec
8 15:18:25	EXPLAIN FORMAT=JSON select * from user	OK	0.000 sec
9 15:18:28	select * from book LIMIT 0, 1000	3 rows returned	0.000 sec / 0.000 sec
10 17:14:48	select * from user LIMIT 0, 1000	8 rows returned	0.000 sec / 0.000 sec

Object Info Session

34°C Sunny

Search

ENG IN 17:14 11-07-2023

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.