Tender Management API

In this hands-on, you need to create a REST Api in Spring boot, which is used to manage the bidding details and get approved.

Models:

RoleModel:

Field Name	Datatype	Primary Key	Foreign Key	Comments
id	Integer	Yes		autoincrement
rolename	String			Unique

UserModel:

Field Name	Datatype	Primary Key	Foreign Key	Comments
id	Integer	Yes		autoincrement
username	String			
companyName	String			
email	String			Unique
password	String			
role	Integer		Yes	

BiddingModel:

Field Name	Datatype	Primary Key	Foreign Key	Comments
id	Integer	Yes		autoincrement
biddingId	Integer			Unique
projectName	String			projectName is a final string with value “Metro Phase V 2024”
bidAmount	Double			
yearsToComplete	Double			
dateOfBidding	String			Current date in dd/MM/yyyy format
status	String			Default value = “pending”
bidderId	Integer		Yes	

If any of the above validations are failing, return with a response code of 400 - Bad Request

We have initialized the database with the following data

Role

id	rolename
1	BIDDER
2	APPROVER

User

username	companyName	password	email	role
bidder1	companyOne	bidder123$	bidderemail@gmail.com	1
bidder2	companyTwo	bidder789$	bidderemail2@gmail.com	1
approver	defaultCompany	approver123$	approveremail@gmail.com	2

Implement JWT based authorization and authentication with the two above mentioned roles. BIDDER and APPROVER should be identified from the JWT token

JWT token should be sent as a Bearer token in Authorization request header. For example: Authorization value would be Bearer <SPACE><JWT TOKEN>

End-Points Marked in

- Red is accessible only by bidders
- Blue is accessible only by approver
- Green is accessible by both bidder and approver

All other endpoints except **/login** should be authenticated and authorized with the above conditions

Endpoints:

1.POST METHOD - **/login**

Authenticates and creates JWT token with respective authorization

Request Parameters	Success Response	Error Response

JSON Body -
{
"email":"bidderemail@gmail.com",
"password":"bidder123$"
}
	200 OK
	{
	"jwt":"your_jwt_token",
	"status":200
	}
		400 Bad Request on invalid credentials

2.POST METHOD - **/bidding/add**

Adds a new Bidding. Note: **bidderId** should point to the bidder who created the bidding.

Request Parameters	Success Response	Error Response

JSON Body -
{
"biddingId":2608,
"bidAmount":14000000.0,
"yearsToComplete":2.6
}
	201 CREATED
	{
	"id": 1,
	"biddingId": 2608,
	"projectName": "Metro Phase V 2024",
	"bidAmount": 1.4E7,
	"yearsToComplete": 2.6,
	"dateOfBidding": "07/07/2023",
	"status": "pending",
	"bidderId": 1
	}
		400 Bad Request on invalid credentials

3.GET METHOD - **/bidding/list**

To get all the details of the bidding which are greater than the **bidAmount** which will be given in the request param and return response with status code 200.

If no data available for given value, then return “no data available” as a response with status code 400.

E.g., /bidding/list?bidAmount=15000000

4.PATCH METHOD - **/bidding/update/{id}**

Updates the bidding **status**.

Request Parameters	Success Response	Error Response

JSON Body -
{
"status":"approved"
}
	200 OK
	{
	"id": 1,
	"biddingId": 2608,
	"projectName": "Metro Phase V 2024",
	"bidAmount": 1.4E7,
	"yearsToComplete": 2.6,
	"dateOfBidding": "07/07/2023",
	"status": "approved",
	"bidderId": 1
	}
		400 Bad Request on invalid credentials

5.DELETE METHOD - **/bidding/delete/{id}**

Note: This Method requires authentication. User who was authenticated and have role "**Approver**" and "**Bidder**" who is also the creator of the given id object, they can able to access this endpoint.

Get the bidding object with given id from bidding detail model, delete it and return "deleted successfully" with status code 204.

If the given id is not found, then return "not found" with status code 400.

If the authenticated user is not a creator of given id object, then return "you don't have permission" with status code 403.

Instructions

- Install the required dependencies by running '**bash install.sh**' from the project folder.
- For running the application use '**mvn spring-boot:run**'.
- For testing the application use '**mvn clean test**'.
- Enable Swagger 3 API Documentation at /v3/api-docs.
- If you are getting port already in use error, open terminal and execute '**fuser -k 8080/tcp**' or "**sudo service jenkins stop**". If you find difficult doing this, go to application.properties and change server.port=8080 to any other port. E.g., server.port=8082
- After completing the hands-on, submit the test.