Faculty Profile Management

This API is used in college where it can perform various operations with details of faculty in the college.

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			
email	String			Unique
password	String			
role	Integer		Yes	

FacultyModel:

Field Name	Datatype	Primary Key	Foreign Key	Comments
id	Integer	Yes		autoincrement
faculty_name	String			
qualification	String			
email	String			Unique
gender	String			
department	String			
experience	Integer			
admin_id	Integer		Yes	

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

Initialize the database with the following data

Role

id	**rolename**
1	ADMIN
2	USER

User

username	**password**	**email**	**role**
admin1	admin123$	adminemail@gmail.com	1
admin2	admin789$	adminemail2@gmail.com	1
user	user123$	useremail@gmail.com	2

Implement JWT based authorization and authentication with the two above mentioned roles. USER and ADMIN 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 admins
- Blue is accessible by both users and admins

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

Endpoints:

1.POST METHOD - **/user/login**

Authenticates and creates JWT token with respective authorization

Request Parameters	**Success Response**	**Error Response**
JSON Body - { "email":"adminemail@gmail.com", "password":"admin123$" }	200 OK { "jwt": "your_jwt_token", "status": 200 }	400 Bad Request on invalid credentials

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

Adds a new faculty. Note: admin_id should point out the id of the user who created the faculty.

Request Parameters	**Success Response**	**Error Response**
JSON Body - { "faculty_name":"facultyOne", "qualification" :"B.E", "email" :"faculty1@gmail.com", "gender":"male", "department":"ECE", "experience":3 }	201 CREATED { "id": 1, "faculty_name": "facultyOne", "qualification": "B.E", "email": "faculty1@gmail.com", "gender": "male", "department": "ECE", "experience": 3, "admin_id": 1 }	400 Bad Request on invalid credentials

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

Get Method:

Note: This Method requires authentication. User who was authenticated can access this endpoint.

Get the department or experience or gender as a parameter from the URL and return faculty details which relates to the given search value with the status code 200.

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

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

Updates the faculty’s qualification , experience and department.

Request Parameters	**Success Response**	**Error Response**

JSON Body -
{
"qualification": "M.E",
"experience": 5 ,
"department": "IT"
}
	200 OK
	{
	"id": 1,
	"faculty_name": "facultyOne",
	"qualification": "M.E",
	"email": "faculty1@gmail.com",
	"gender": "male",
	"department": "IT",
	"experience": 5,
	"admin_id": 1
	}
		400 Bad Request on invalid credentials

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

Note: This Method requires authentication. User who was authenticated and has role “ADMIN” as well as the creator of the given id object, they can access this endpoint.

Get the faculty object with given ID from faculty detail model, delete it and return “deleted successfully” with status code 204.

If the given id 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**’.
- After completing the hands-on, submit the test.