TestCase with API Signature

Task Description

Development	Perform CRUD Operation	1. The project should have a process of user registration, updation, deletion, and retrieval.
		2. The project should consider the unique username even after the deletion of one user the same should not be used.
		3. After registration a password should be generated (need not implement the sms part, return the password in response). This password should be directly linked with the user name. i.e. username can be retrieved via password. Ex. userName: ANURAG, password: 114211817 Note: This password mapping should reflect the strong encryption, logic behind the encryption should have a strong base. The retrieval should not be ambiguous (like the given example).
		4.Log Level and effectiveness of logs.

INDEX

*Note:The project runs on port 8080, and the API testing tool Postman is open.

Test Case 1: User Registration

Summary: To verify that the successfully registered with a unique username.

Given: A user wants to register with a unique username along with other details.

When: The user sends a POST request to /api/users/register/{username} with the following JSON body:

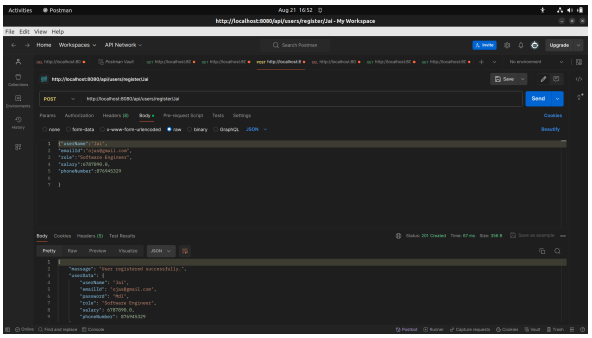
{
"userName": "username",
"emailId": "email",
"role": "role",
"salary": salary,
"phoneNumber":phoneNumber
}

*All field are mandaory

Then: The API should respond with a status code of `201 Created` and the following JSON body:

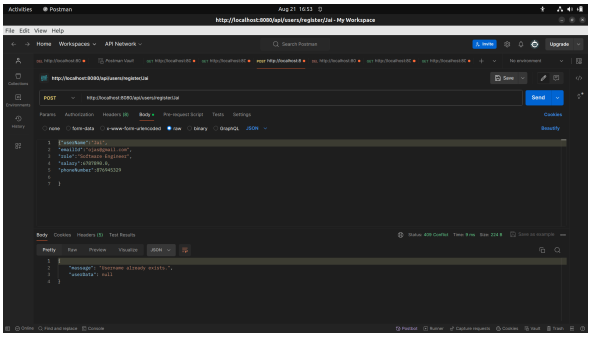
{ "message": "User registered successfully",
userData: {
"userName": "username",
"emailId": "email"
"password": "encrypted_password"
"role": "String",
"salary": salary,
"phoneNumber": phoneNumber
}
}
Result: passed

Screenshot:



Test Case 2: Duplicate User Registration

Summary: To verify that the user registers with an existing username.	

Given: A user tries to register with a username that already exists in the system.
 	
When: The user sends a POST request to /api/users/register/{username}
with the following JSON body:
 	
{
"userName":"username"
"emailid": "email",
"role": "role",
"salary": salary,
"PhoneNumber": phoneNumber
}
 	
Then: The API should respond with a status code of 400 Bad Request and the following JSON body:
{ "message": "User already registered"
userData:null
 	
}
 	
Result: passed
	Screenshot: to be pasted
	 

Test Case 3: User Retrieval(getting user details)

Summary: To verify the registered user wants to view their account details with an existing username.

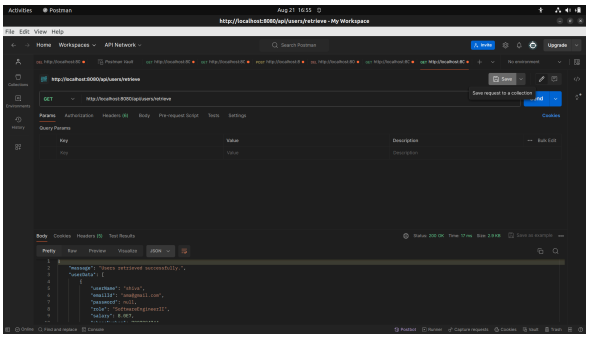
Given: A registered user wants to view their account details.

When:The user sends a GET request to /api/users/retrieve/{username}

Then: The API should respond with a status code of 200 OK and the following JSON body:
{ "message": "User retrieved successfully",
userData:{"userName": "username",
 “emailId": "email”
 "password": "null"
 "role": "String",
  "salary": salary,
  "phoneNumber": phoneNumber
}
 }

Result: passed

Screenshot:

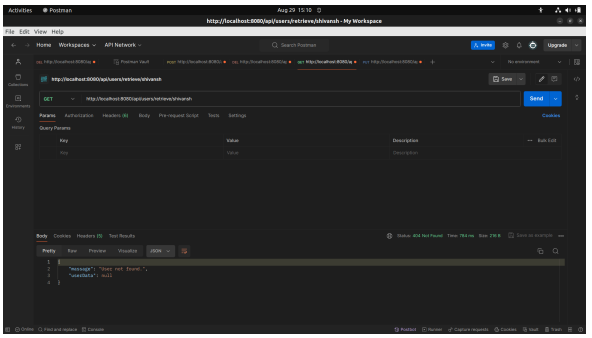


Test Case 4: User Retrieval of Non-Existing Username

<table>
<tr><td colspan="2">Summary: To verify the registered user wants to view their account details with a non-existing username.</td></tr>
<tr><td>Given: A user tries to retrieve details for a username that does not exist in the system.

When: The user sends a GET request to /api/users/retrieve/{username}.

Then: The API should respond with a status code of 404 Not Found and the following JSON body:
{“message”:"User does not found"
}

Result: passed</td><td>Screenshot:
  </td></tr>
</table>

Test Case 5: User Updation

Summary: To verify that the user has updated their account with an existing username.

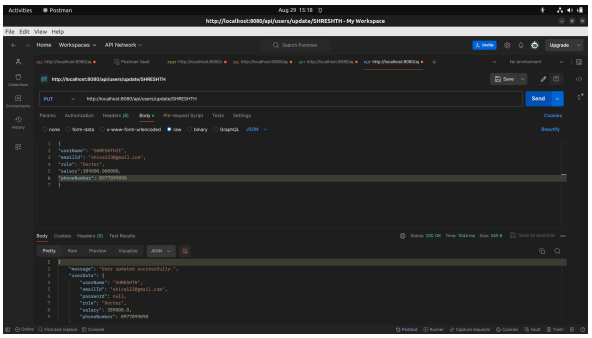
Given: A registered user wants to update their account details.

When: The user sends a PUT request to /api/users/update/{username} with the following JSON body:
{
“username”:”username”
"emailId": "email",
"role": " role",
"salary": salary,
"phoneNumber": phoneNumber
}

Then: The API should respond with a status code of 200 OK and the following JSON body:
{
“message”:” User data updated successfully”
userData:{
“username”:” username”(can’t updated)
"email": "email", (updated)
"role": "role", (updated)
"salary": salary, (updated)
"phoneNumber": phoneNumber (updated)
}
}

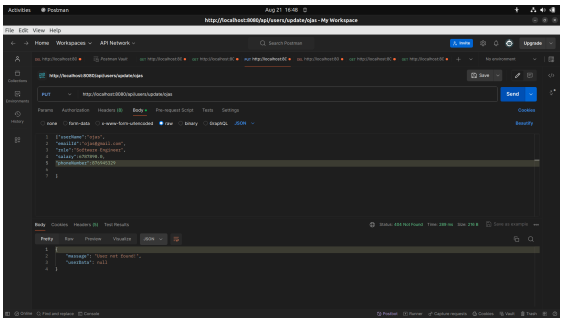
Result: passed

Screenshot:

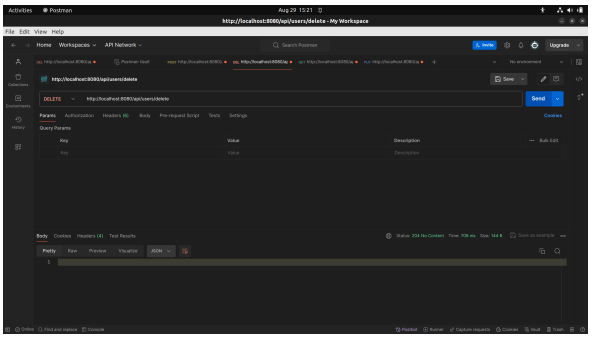


Test Case 6: User Updation of Non-Existent User

Summary: To verify that the user has updated their account with a non-existing username.	

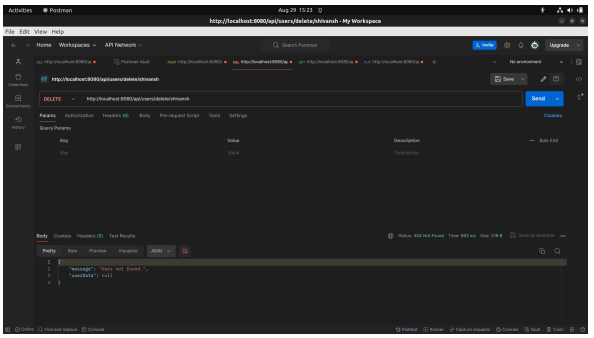
Given: A user tries to update the details of a user that does not exist.
 	
When: The user sends a PUT request to /api/users/update/{username} with the following JSON body:
 	
{
“username”:”username”
"emailId": "email",
"role": " role",
"salary": salary,
"phoneNumber": phoneNumber
}
 	
Then: The API should respond with a status code of `404 Not Found` and the following JSON body:
 	
{"message": "user does not found”
}
 	
Result:
	Screenshot: To be pasted
	
	 

Test Case 7: User Deletion

Summary: To verify the registered user deleted their account.	
Given: A registered user wants to delete their account. **When**: The user sends a DELETE request to /api/users/delete/{username}. **Then**: The API should respond with a status code of 200 OK and with a blank JSON body. **Result**: Passed	Screenshot: 

Test Case 8: User Deletion of Non-Existent User

Summary: To verify the registered user deleted their account that does not exist	

Given: A user registered wants to delete a username that does not exist.
 	
When: The user sends a DELETE request to /api/users/delete/{username}.
 	
Then: The API should respond with a status code of 404 Not Found and the with blank json body:
{
 	
"massage": "User not found.",
"userData": null
 	
}
 	
Result: passed
	Screenshot:
	 

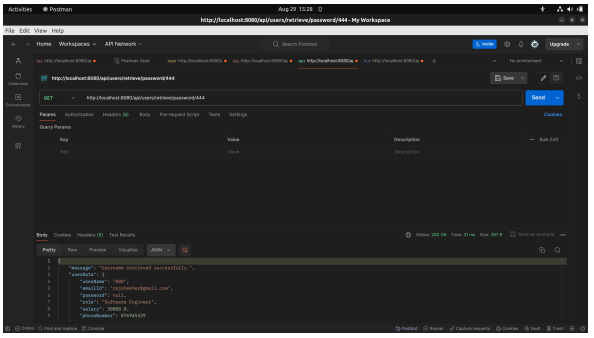
Test Case 9: Retrieve after Deleted User

<table>
<tr><td colspan="2">Summary: To verify that the user has deleted their account yet old username existing in db.</td></tr>
<tr><td>Given: A user want to retrieve by password which user has been deleted from the system.

When:A request is made to retrieve user by (password) sending a GET request to /api/users/retrieve/password/{password}

Then: The API should respond with a status code of 200 OK and the following JSON body:

{ "message": "User retrieve successfully",
userData:{"username": "username",
“emaiidl": "email",
"password": null
"role": "role",
"salary": salary,
"phoneNo":phonenumber
}
}

Result:passed</td><td>Screenshot:
  </td></tr>
</table>

API Signature List

CRUD Operation	Method	URL End-Points	Parameter
1. User Registration	POST	/api/users/register/{username}	Request json body: { "username": "String",(mandatory) "email": "String", "role": "String", "salary": double, "phoneNo": int } Response Json Body: { "message": "User registered successfully", "username": "string", "password": "encrypted_password" “email": "String", "role": "String", "salary": double, "phoneNo": int }
2. User Registration with an existing account(username)			

3. User Retrieve/Retrieve-by Id	GET	/api/users/retrieve
		/api/users/retreive/{username}
			No Needed request body Json
			
			Response body Json:
			{
			"username": "string",
			"password": "encrypted_password"
			“email": "String",
			"role": "String",
			"salary": double,
			"phoneNo": int
			
			}

4. User Retrieve non-existing account			

5. User Updation 6. User Updatation with a non-existing account	UPDATE	/api/users/update/{username}	Updated request body Json {"email": "String", "role": "String", "salary": double, "phoneNo": int } Response Body Json: {“message”:” User data updated successfully” “username”:” string”(can’t updated) "email": "String", updated "role": "String", updated "salary": double, updated "phoneNo": int }
7. User Deletion/Delete-byId 8. User Deletion with a non-existing account	DELETE	/api/users/delete /api/users/delete/{username}	No Needed Response body Json: {“message”:” user successfully deleted:}
9. User Retrieve using password as username	GET	/api/users/retrieve?password=value	NoNeeded Request body Json Response body Json: { "message": "User login successfully", "username": "string", //"password": "encrypted_password" “email": "String", "role": "String", "salary": double, "phoneNo": int }

After Implementation

API Signature List

CRUD Operation	Method	URL End-Points	Parameter output

1. User Registration	POST	/api/users/register/{username}
		
		http://localhost:8080/api/users/register/Himanshu
			Request json body:
			`{"userName":"Himanshu", "emailId":"Him@gmail.com", "role":"SystemAdmin", "salary":500000.0, "phoneNumber":909009009 }`
			
			Response Json Body:
			`{ "massage": "User registered successfully.", "userData": { "userName": "Himanshu", "emailId": "Him@gmail.com", "password": "Klpdqvkx", "role": "SystemAdmin", "salary": 500000.0, "phoneNumber": 909009009 } }`
			
			2. Response Json Body for duplicate user
			`{ "massage": "Username already exists.",`

2. User Registration with an existing account(username)			

			<pre>{
			"userData": null
			}</pre>

3. User Retrieve/Retrieve-by Id
 	
4. User Retrieve non-existing account
	GET	/api/users/retrieve
		/api/users/retreive/{username}
		http://localhost:8080/api/users/retrieve
			No Needed request body Json
			
			Response body Json:
			<pre>{
			"massage": "Users retrieved successfully.",
			"userData": [
			{
			"userName": "shiva",
			"emailId": "shivaJi@gmail.com",
			"password": null,
			"role": "Manager",
			"salary": 8.0E7,
			"phoneNumber": 654678345
			},
			{
			"userName": "shivam",
			"emailId": "ram123@gmail.com",
			"password": null,
			"role": "Api Developer",
			"salary": 50000.0,
			"phoneNumber": 7896782340
			},
			//////
			////////
			
			{</pre>

<table>
<tr>
<td></td>
<td></td>
<td></td>
<td>
<pre>
"userName": "Himanshu",
"emailId": "Him@gmail.com",
"password": null,
"role": "SystemAdmin",
"salary": 500000.0,
"phoneNumber": 909009009
}
]
}
</pre>

<pre>
{
"massage": "User retrieved successfully.",
"userData": {
"userName": "dipika",
"emailId": "deepikah@gmail.com",
"password": null,
"role": "SoftwareEngineerII",
"salary": 8.0E7,
"phoneNumber": 7007894735
}
}
</pre>
</td>
</tr>
<tr>
<td>5. User Updation

6. User Updatation with a non-existing account</td>
<td>UPDATE</td>
<td>/api/users/update/{username}

http://localhost:8080/api/users/update/dipika</td>
<td>Updated request body Json
<pre>
{
"userName": "dipika",
"emailId": "deepu123@gmail.com",
"password": null,
"role": "SoftwareEngineer",
</pre>
</td>
</tr>
</table>

			<pre>"salary": 25.0E7,
			"phoneNumber": 7070777899
			}</pre>
			<pre>{
			"massage": "User updated successfully.",
			"userData": {
			"userName": "dipika",
			"emailId": "deepu123@gmail.com",
			"password": null,
			"role": "SoftwareEngineer",
			"salary": 2.5E8,
			"phoneNumber": 7070777899
			}
			}</pre>

7. User Deletion/Delete-byId 8. User Deletion with a non-existing account	DELETE	/api/users/delete /api/users/delete/{username} http://localhost:8080/api/users/delete http://localhost:8080/api/users/delete/skri	No Needed Response body Json: Blank

9. User Retrieve using password as username	GET	/api/users/retrieve?password=value
		
		*I have replaced `?password=value` with `{password}` so that all APIs follow a similar format.
		and I changed the path to `/api/users/retrieve/password/{password}` to avoid ambiguity between `/api/user/retrieve/{username}` and `/api/user/retrieve/{password}`.
		
		http://localhost:8080/api/users/retrieve/password/787
			NoNeeded Request body Json
			
			Response body Json:
			{
			"massage": "Username retrieved successfully.",
			"userData": {
			"userName": "454",
			"emailId": "ama@gmail.com",
			"password": null,
			"role": "SoftwareEngineerII",
			"salary": 8.0E7,
			"phoneNumber": 7007894744
			}
			}
			*Ye user Jo db storage user hai