

International University
School of Computer Science and Engineering

**Web Application Development
Laboratory
IT093IU
Lab #9**

**Submitted by
Đặng Nguyễn Nhật Vy - ITITDK22102**

Current user profile

The screenshot shows the Postman interface with a collection named "jwt_authentication". A GET request is made to `/api/users/profile`. In the Headers tab, there is an Authorization header set to `Bearer eyJhbGciOiJIUzUxMjQ.... The response status is 200 OK, and the response body is a JSON object:`

```

1 {
2   "id": 4,
3   "username": "testuser",
4   "email": "test@example.com",
5   "fullname": "Test User",
6   "role": "USER",
7   "isActive": true,
8   "createdAt": "2025-12-13T15:40:44"
9 }

```

-> This test verifies the functionality of retrieving the currently authenticated user's profile. The request includes a valid JWT token in the Authorization header. Spring Security validates the token, extracts the user identity, and stores it in the SecurityContext. The controller then accesses the authenticated user information and returns a UserResponseDTO containing the user's profile details. A successful 200 OK response confirms that the application correctly identifies the current user based on the JWT token and returns accurate profile data.

Change password success

The screenshot shows the Postman interface with a collection named "jwt_authentication". A PUT request is made to `/api/change-password`. In the Body tab, the raw JSON payload is:

```

1 {
2   "currentPassword": "password123",
3   "newPassword": "12345pp",
4   "confirmPassword": "12345pp"
5 }

```

The response status is 200 OK, and the response body is a JSON object:

```

1 {
2   "message": "Password changed successfully"
3 }

```

-> This test validates the change password feature for an authenticated user. The request includes the current password, a new password, and its confirmation. The controller retrieves the current user from the SecurityContext, verifies that the provided current password matches the stored hashed password, and checks that the new password and confirmation are identical. If all validations pass, the new password is hashed using BCrypt and saved to the database. A successful response confirms that the password update process works correctly and securely.

Login with new password

The screenshot shows the Postman application interface. The left sidebar displays collections like 'customer_api' and 'jwt_authentication'. The main area shows a POST request to `/api/auth/login` at `http://localhost:8080/api/auth/login`. The request body is set to raw JSON:

```
1
2 {
3     "username": "testuserx",
4     "password": "12345PP"
5 }
```

The response status is 200 OK, with a duration of 194 ms and a size of 752 B. The response body is a JSON object containing an access token and refresh token.

-> This test confirms that the password change has taken effect. The user attempts to log in using the newly updated password. Spring Security authenticates the credentials against the stored hashed password in the database. Since the password was successfully changed in the previous step, authentication succeeds and a new JWT access token is generated and returned. This demonstrates that the password update is persistent and that the user can continue to authenticate normally using the new credentials.