

Bcrypt Encoding For User Registration

- When we implement Bcrypt in an existing application, we face a challenge:
- Existing users already have plain text passwords in the database
 - Our authentication system expects plain text passwords
 - Once Bcrypt is implemented, the system will try to compare Bcrypt hashes with plain text

➤ Update Existing Password Data

Before implementing Bcrypt, we need to:

- Update existing passwords in the database
- Convert plain text passwords to Bcrypt-encoded versions
- This can be done using an update query after generating encoded passwords

You can generate encoded passwords using an online tool like

[https://www.browsrling.com:](https://www.browsrling.com/)

The screenshot shows a web page titled "Bcrypt Password Generator" under the "cross-browser testing tools" category. The page claims to be "World's simplest online bcrypt hasher for web developers and programmers". It features a text input for "Password" containing "n@345", a dropdown for "Rounds" set to "12", and two buttons: "Bcrypt" and "Copy to clipboard". Below the input fields is a large text area displaying the hashed output: "\$2a\$12\$RH4dw5t.8kR7wQj/Eh4u6udsytqhpqm39xW.GFeEl9mUHUeDYM26". At the bottom, there's a note about testing with the "Bcrypt Hash Tester" tool.

The screenshot shows a MySQL Workbench interface. In the top-left corner, there's a 'Query' tab and a 'Query History' tab. Below them is a code editor containing the following SQL queries:

```

1 SELECT * FROM users;
2
3 update users set password='$2a$12$RH4dWSt.8kR7wQj/Eh4u6udsytqhpqm39xW.GFeEW9mUHueDYM26' where id=3
4

```

Below the code editor is a 'Data Output' tab, which is currently selected. It displays a table with three rows of data:

	id [PK] integer	username text	password text
1	1	kiran	n@789
2	2	harsh	h@123
3	3	navin	\$2a\$12\$RH4dWSt.8kR7wQj/Eh4u6udsytqhpqm39xW.GFeEW9mUHueDYM26

➤ Implementing Bcrypt in the Service Layer

- In the `UserService` class, create a `BCryptPasswordEncoder` object
- Set the strength parameter (rounds) to 12 for good security
- Before saving the user, encode the password using the encoder
- Save the user with the encoded password to the database

Example:

```

● ● ●

package com.telusko.springsecdemo.service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;
import com.telusko.springsecdemo.dao.UserRepo;
import com.telusko.springsecdemo.model.User;

@Service
public class UserService {

    @Autowired
    private UserRepo repo;
    private BCryptPasswordEncoder encoder = new BCryptPasswordEncoder(12);

    public User saveUser(User user) {
        user.setPassword(encoder.encode(user.getPassword()));
        System.out.println(user.getPassword());
        return repo.save(user);
    }
}

```

Output:

The screenshot shows a Postman interface with a POST request to `localhost:8080/register`. The request body is a JSON object with fields `id`, `username`, and `password`. The response is a 200 OK status with a JSON object containing the same fields and a hashed password.

```
POST /register
{
  "id": 4,
  "username": "avani",
  "password": "a@123"
}

{
  "id": 4,
  "username": "avani",
  "password": "$2a$12$TaMuHwsZerp.O5h9oiwzieX6tqsgk1PU.sGsu0nP6vnjV7Miunv2m"
}
```

The screenshot shows a DBeaver interface with a SQL query window running the command `SELECT * FROM users;`. The results pane displays four rows of data from the `users` table, showing columns `id`, `username`, and `password`.

	<code>id</code> [PK] integer	<code>username</code>	<code>password</code> text
1	1	kiran	n@789
2	2	harsh	h@123
3	3	navin	\$2a\$12\$SRH4dWSt.8kR7wQj/Eh4u6udsytqhpmqm39xW.GFeEW9mUH... ...
4	4	avani	\$2a\$12\$TaMuHwsZerp.O5h9oiwzieX6tqsgk1PU.sGsu0nP6vnjV7Miunv2m

👉 Remember

- **BCryptPasswordEncoder** is part of Spring Security framework
- The parameter **12** represents the work factor ($2^{12} = 4,096$ rounds)
- Higher work factor means better security but slower performance
- **encode()** method transforms the plain text password into a Bcrypt hash
- We're printing the encoded password to console for demonstration (not recommended in production)
- After this implementation, all new users will have encoded passwords
- The authentication system must also be updated to verify with Bcrypt (covered in next section)