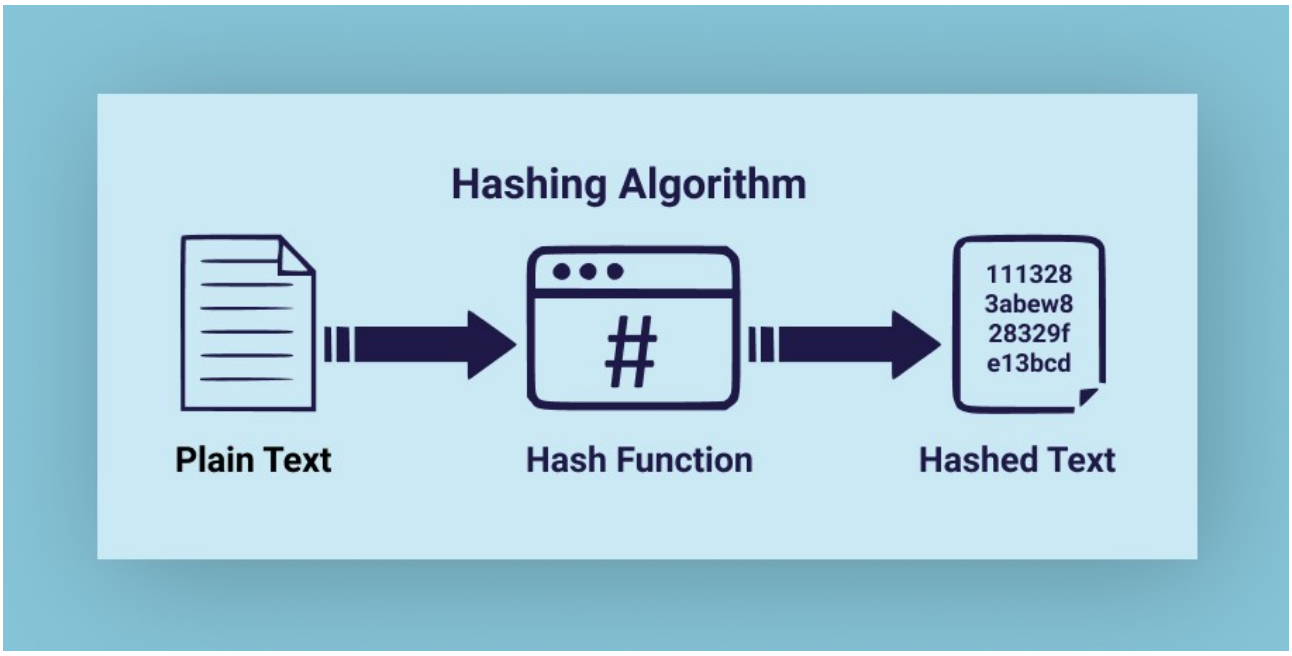
Creating a Register and Login Application

in this application we will demonstrate the use of hashing algorithms.

Cryptography is the process of hiding or coding information so that only the person a message was intended for can use it. The art of cryptography has been used to code messages for thousands of years and continues to be used in bank cards, computer passwords, and ecommerce.

What is Hashing?

Hashing is a data security technique used in Cryptography to convert data values into alternate, unique identifiers called hashes for quick and secure access. Hashing can be used for data security because the one-way process prevents access to or tampering with the source data. In simple Terms, Hashing is the process of transforming any given key or a string of characters into another value.



Some common hashing algorithms include MD5, SHA-1, SHA-2, NTLM, and LANMAN, Bcrypt etc

Practical.

Create a Flask Project. Create **templates** Folder and an **app.py**

Create a **functions.py** and put the code in below link.

https://justpaste.it/f1bvw

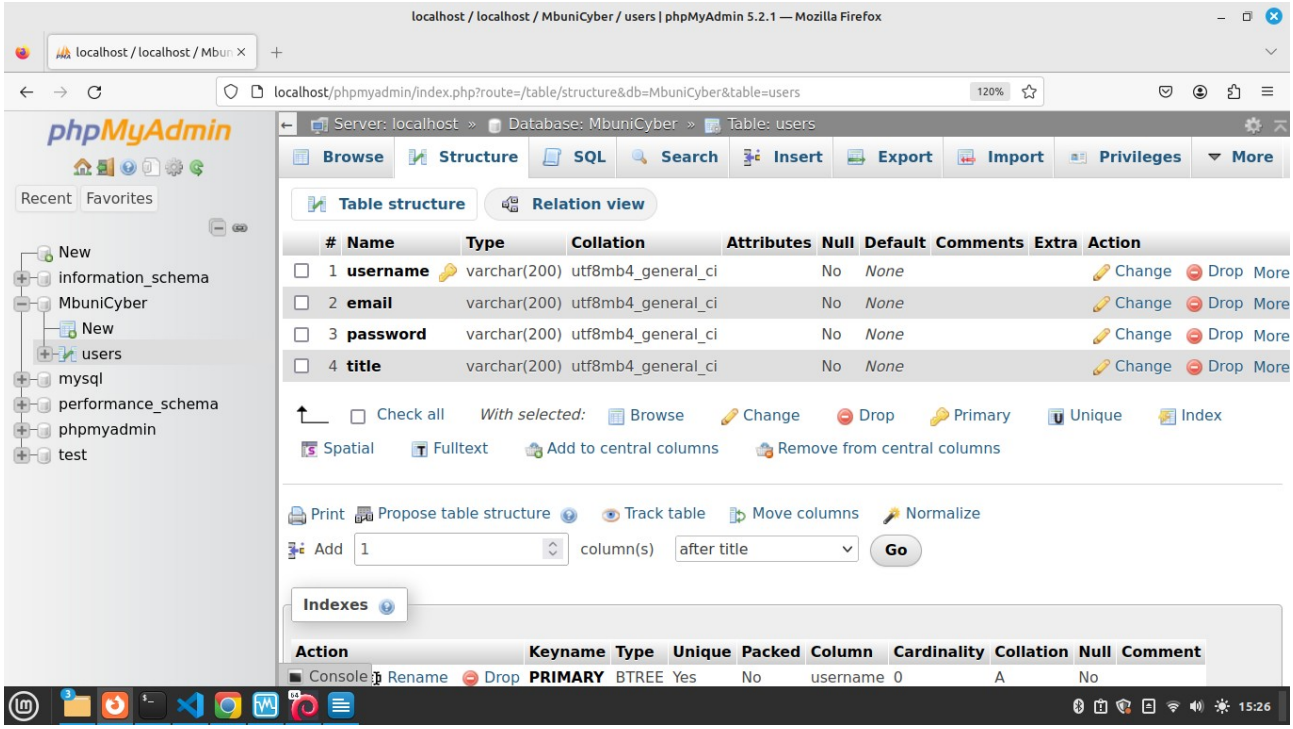
Above explain **hash_password()** Function.

In the above code, we are hashing a password using the MD5 algorithm.

The hash_password function takes a password as input, creates a new MD5 hash object using hashlib.md5(), then hashes the password using the update() method of the hash object, and finally gets the hexadecimal representation of the hash using hexdigest().

MD5 is considered to be a weak hashing algorithm because it has known vulnerabilities and is susceptible to collision attacks, where two different inputs produce the same hash output. This makes it easier for attackers to reverse-engineer the hashed password, especially if they have access to the hashed values.

In **Xampp**, Create a database named **YourClassCyber**, create a table named **users** with columns as shown below.



In Your Flask templates Folder, create a file named **signup.html** and write below code.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
</head>
<body>
<h1>Sign Up Application</h1>
{{msg}}
<form action="/signup" method="post">
<input type="text" name="username" placeholder="Enter Your Username"> <br><br>
<input type="email" name="email" placeholder="Enter Your Email"> <br><br>
<input type="password" name="password" placeholder="Enter Your Password">
<br><br>
<input type="text" name="title" placeholder="Enter Job Title"> <br><br>
<input type="submit" value="Make Application">
</form>
</body>
</html>
```

https://justpaste.it/chjeo

in app.py create the Flask app and the signup Route.

```python
from flask import *
import pymysql
app = Flask(__name__)

from functions import *
# Routes Here



app.run(debug=True)
```

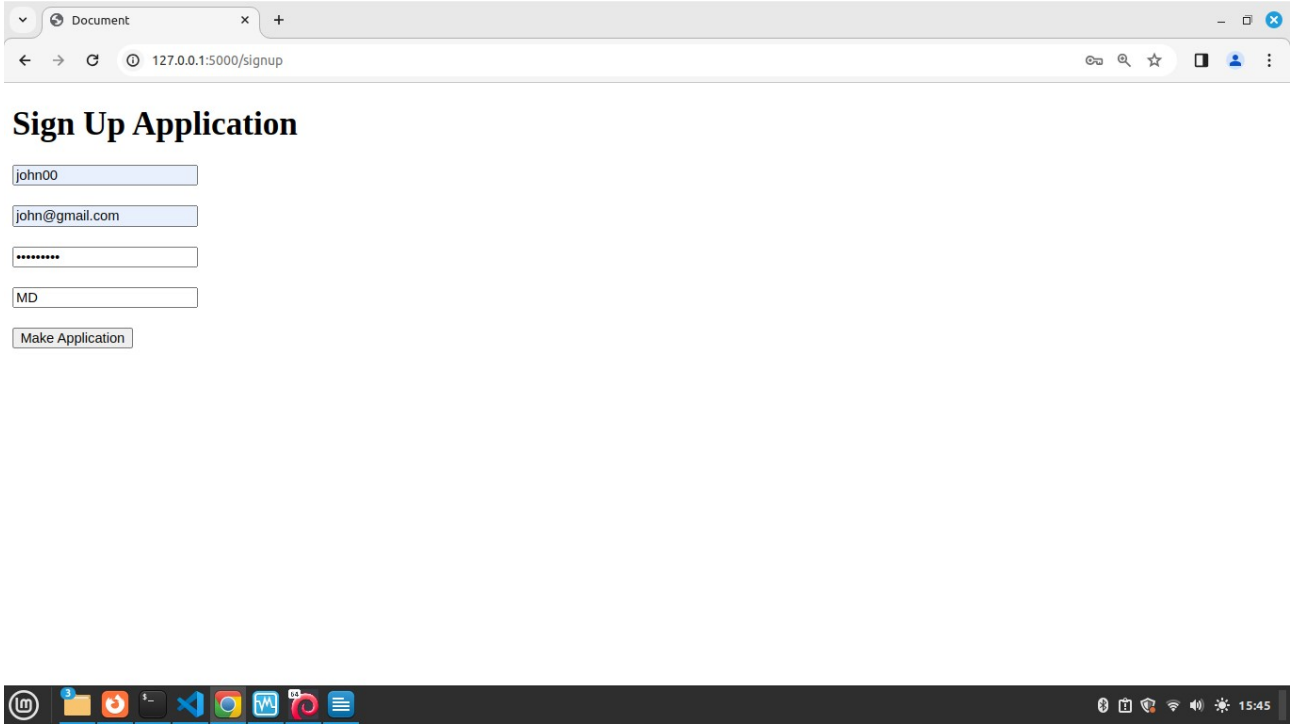
Add the signup route. This rout will receive values from the form and save them to our users table. While saving the password we hash it, meaning its scrambled.

See next page.

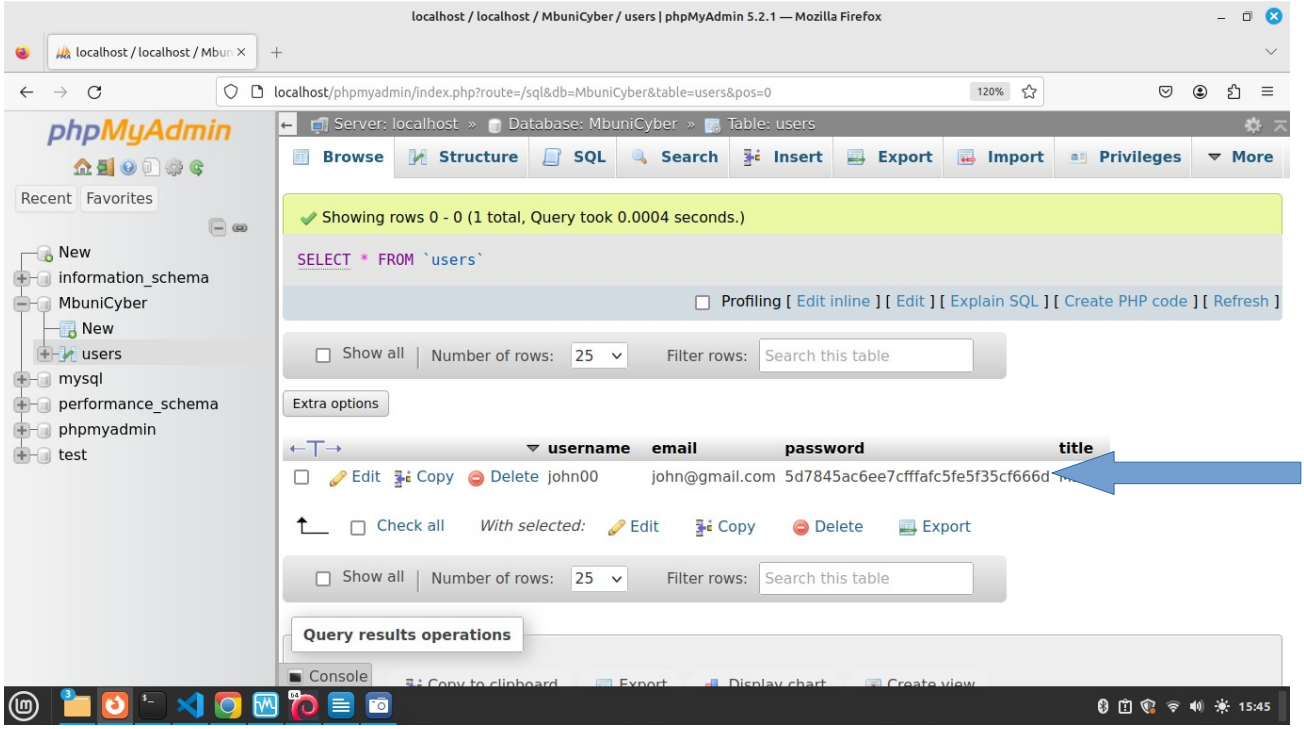
```python
6   # Routes Here
7   @app.route('/signup', methods=['POST', 'GET'])
8   def signup():
9       # Check if form was posted by user
10      if request.method == 'POST':
11              # Receive what was posted by user including username, password1,password2 email, phone
12              username = request.form['username']
13              email = request.form['email']
14              password = request.form['password']
15              title = request.form['title']
16              # Now we can save username, password, email, phone into our users table
17              # Make a connection to database
18              connection = pymysql.connect(host='localhost', user='root', password='',
19                                           database='MbuniCyber')
20              # Create an Insert SQL, Note the SQL has 4 placeholders, Real values to be provided later
21              sql = '''
22                      insert into users(username, email, password, title)
23                      values(%s, %s, %s, %s)
24                  '''
25              # Create a cursor to be used in Executing our SQL
26              cursor = connection.cursor()
27              # Execute SQL, providing the real values to replace our placeholders
28              cursor.execute(sql, (username, email, hash_password(password), title))
29              # Commit to Save to database
30              connection.commit()
31              # Return a message to user to confirm successful registration.
32              return render_template('signup.html', msg='Application Made Successfully')
33      else:
34          # Form not posted, display the form to allow user Post something
35          return render_template('signup.html')
36
```

Please Note on above code line number 28, we hash the password.

Run your Application and access http://127.0.0.1:5000/signup a Form appears Fill in details and submit. NB: use **secret123** as your password, we make the password simple!



In our database the password appears in a secure hash

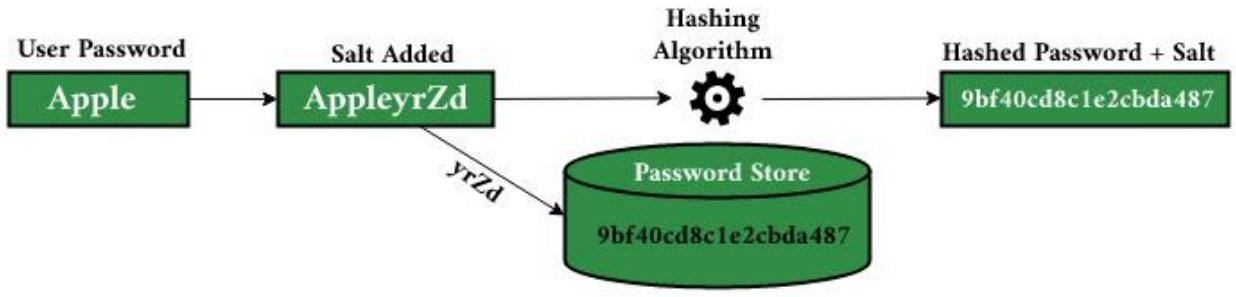


Can we Crack above stored hash?

Please Check PDF 04a - **05a_JtR Password Cracking.pdf**

From above practical we see that we can crack the hashed password! Hence Its insecure. We now improve our hashing approach and add a **salt**

A **salt** is a random value that is used as an additional input to the hashing function to protect against attacks that use precomputed tables to reverse the hashes.



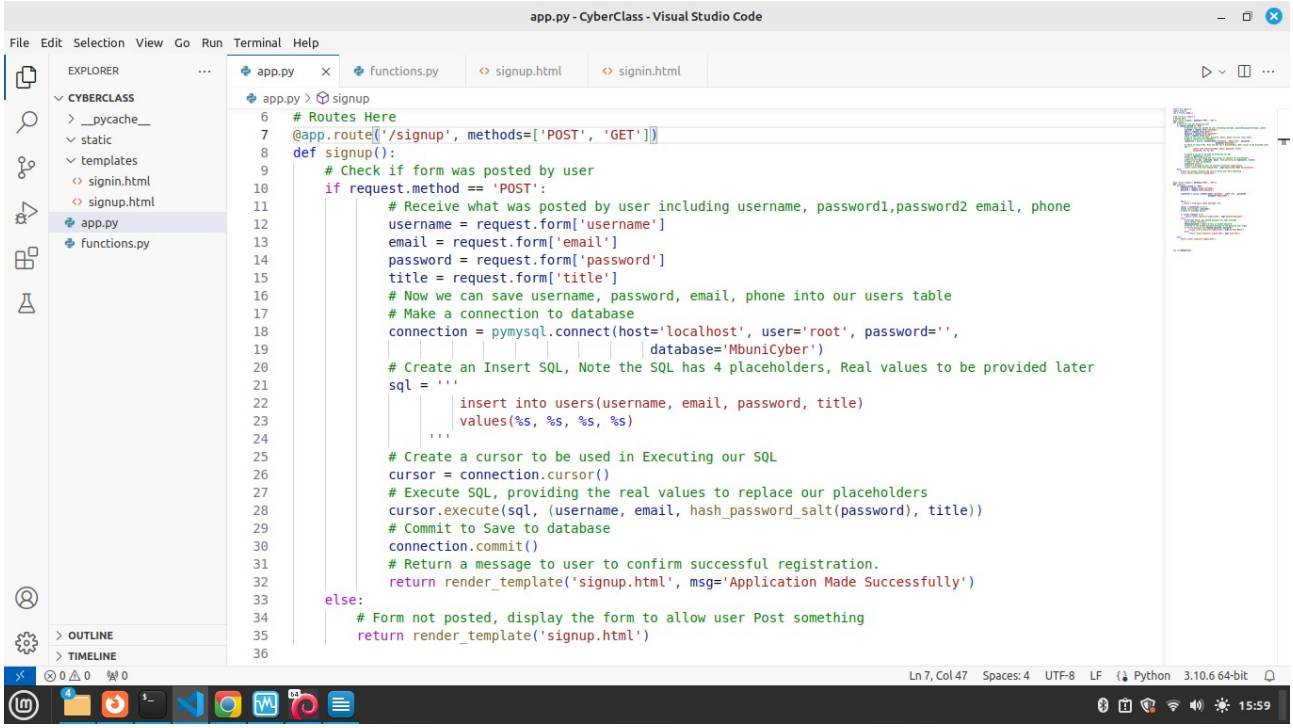
Lets add a salt to the hash and make it stronger.

In your signup route use **the hash_password_salt** function.

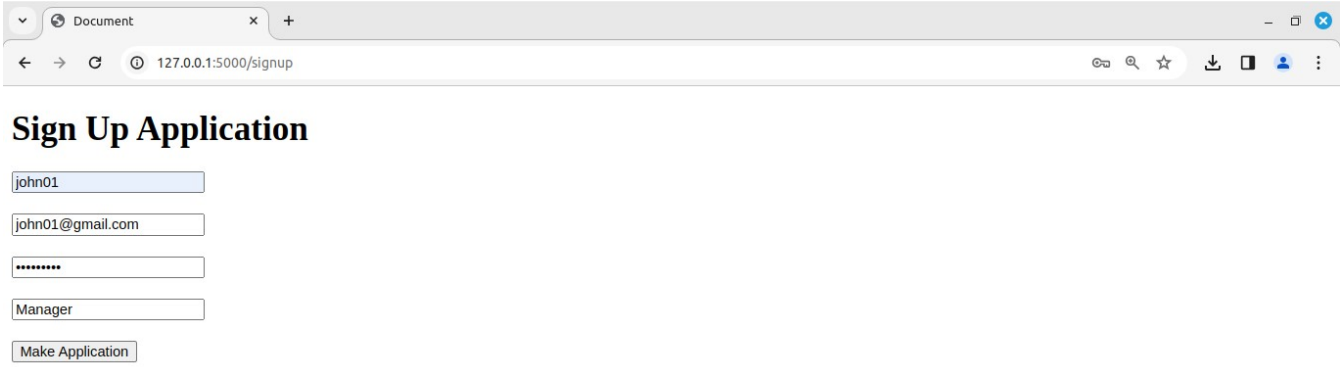
https://justpaste.it/f1bvw

Above explain **hash_password_salt()** Function.

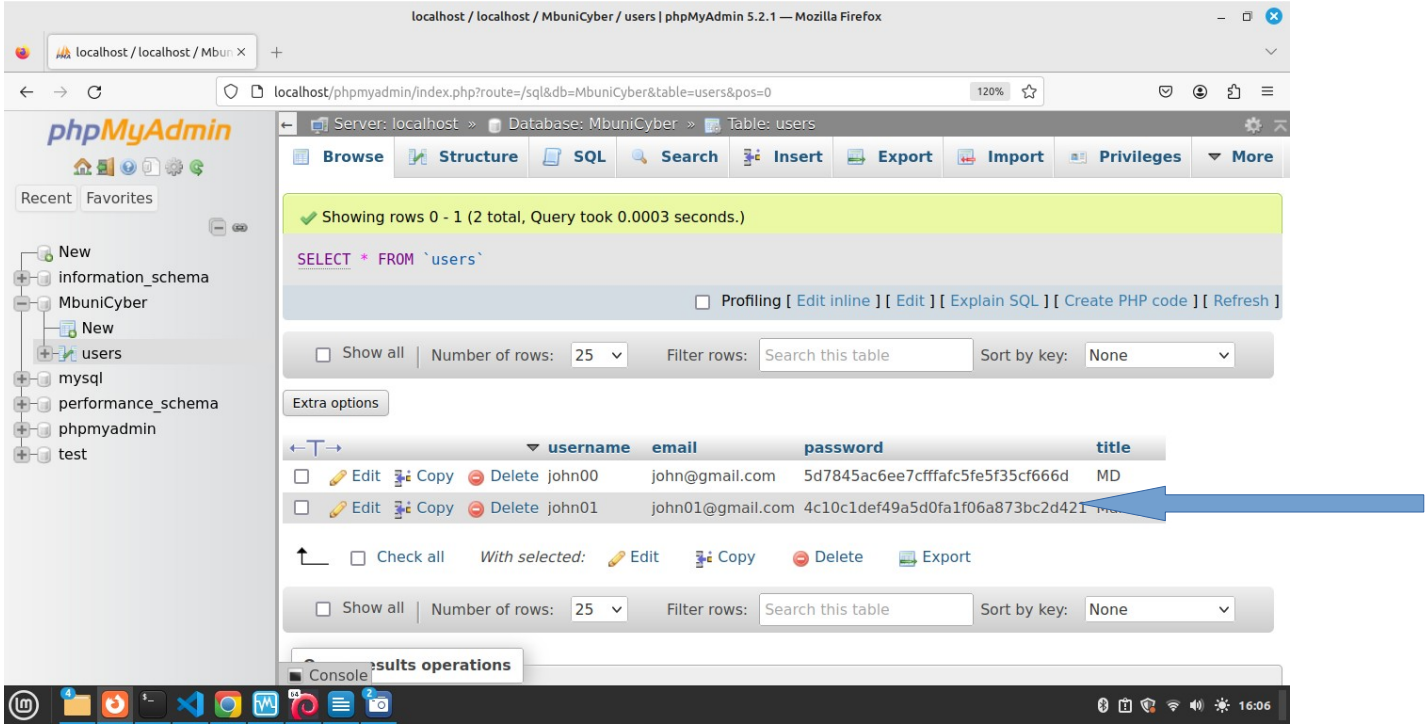
Check line 28 below.



Then run your application and access http://127.0.0.1:5000/signup



We notice a different salted hash is generated. This cannot be be cracked since its salted. You may want to try and crack it using JTR



Other hashing algorithms that use salt in its hashing include Bcrypt, AES, RSA, Blowfish etc.

Useful Link
https://www.geeksforgeeks.org/hashing-passwords-in-python-with-bcrypt/

Implement the login to verify the salted Password.
https://github.com/modcomlearning/CyberClass