Complete the implementation of register in such a way that it allows a user to register for an account via a form.

* Require that a user input a username, implemented as a text field whose name is username. Render an apology if the user’s input is blank or the username already exists.

#for username field blank \_

If not request.form.get(“username”):

Error message : “Please provide a username”

#for if username exists already

#query database first

rows = db.execute("SELECT \* FROM users WHERE username = ?", request.form.get("username"))

if len(rows) == 1 :

Error message: “ Username Exists”

* Require that a user input a password, implemented as a text field whose name is password, and then that same password again, implemented as a text field whose name is confirmation. Render an apology if either input is blank or the passwords do not match.

#if password and confirm password is blank

If not request.form.get(“password”):

Error message : “Please provide a password”

Elif not request.form.get(“confirmation”):

Error message : “Please enter in password again”

#if passwords don’t match

If request.form.get(“password”) != request.form.get(“confirmation”):

Error Message: “Passwords must match”

* Submit the user’s input via POST to /register.

If request.method == ‘POST’:

* INSERT the new user into users, storing a hash of the user’s password, not the password itself. Hash the user’s password with [generate\_password\_hash](https://werkzeug.palletsprojects.com/en/2.3.x/utils/#werkzeug.security.generate_password_hash) Odds are you’ll want to create a new template (e.g., register.html) that’s quite similar to login.html.

Else:

passwordHash <- generate\_password\_hash(user input password)

db.execute(“Insert users (username,hash) Values (username,passwordHash)”)

return render\_template(“register.html”)

Once you’ve implemented register correctly, you should be able to register for an account and log in (since login and logout already work)! And you should be able to see your rows via phpLiteAdmin or sqlite3.

Implementing user to have at least 1 symbol, 1 number ,1 lowercase and 1 uppercase:

Function Checkpass(userpass):

Has\_upper =Check if there is uppercase

Has\_Lower = Check if there is lowercase

Has\_digit =If any (digit in userpass)

Has\_symbol = If any(symbol in userpass)

Return (has\_upper & has\_lower & has\_digit & has\_symbol)

If Checkpass(userpass) != True:

Error Message : “Password requires atleast one digit, one symbol , 1 lower and 1 uppercase character.”