```python
from flask import Flask, render_template, request, url_for, redirect
from flask_mysqldb import MySQL
import MySQLdb
import MySQLdb.cursors
#LIBRARIES FOR BLOCKCHAIN
import datetime                 #Time stamp
import hashlib                  #hash the blocks



app = Flask(__name__)


#configure MySQL database, link it with flask python by configuring, db name,
root name, password and host name for server
app.config["MYSQL_HOST"] = "127.0.0.1"
app.config["MYSQL_USER"] = "root"
app.config["MYSQL_PASSWORD"] = "karan19"
app.config["MYSQL_DB"] = "login"

db = MySQL(app)

#run if POST or GET method present in form tag of HTML5 file
@app.route('/', methods = ['GET', 'POST'])
def login():
    #check method
    if request.method == 'POST':
        #check whether entered username and password is in databse, if present
 open home page else redirect to register page
        if 'username' in request.form and 'password' in request.form:
            username = request.form['username']

            password = request.form['password']
            cursor = db.connection.cursor(MySQLdb.cursors.DictCursor)
            #fire MySQL query to check values
            cursor.execute("SELECT * FROM info WHERE email=%s and psw=%s", (us
ername, password))
            info1 = cursor.fetchone()
            if info1 is not None:
                if info1['email'] == username and info1['psw'] == password:
                    return redirect(url_for('home'))
            else:
                return redirect(url_for('new'))

    return render_template("login.html")

@app.route('/register', methods = ['GET', 'POST'])
def new():
    #check which method is present, if POST and if user has input the followin
g one to five values
    # store the request of form in variables
    if request.method == 'POST':
        if 'one' in request.form and'two' in request.form and 'three' in reque
st.form and 'four' in request.form and 'five' in request.form:
           Firstname = request.form['one']
           Lastname = request.form['two']
           gender = request.form['three']
           username = request.form['four']
           password = request.form['five']
           #access cursor of MySQL
           cur = db.connection.cursor(MySQLdb.cursors.DictCursor)
           #fire MySQL query to write registered data in table info of db logi
n
           cur.execute("INSERT INTO login.info(Fname, Lname, gender, email, ps
w)VALUES(%s, %s, %s, %s, %s)", (Firstname, Lastname, gender, username, passwor
d))
           db.connection.commit()
           return redirect(url_for('login'))

        else:
            return redirect(url_for('new'))

    return render_template("register.html")

@app.route('/home')
def home():
    return render_template("home.html")

#check which method is present
@app.route('/pass', methods = ['GET', 'POST'])
def passdata():

    #START OF BLOCKCHAIN CODE - Skeleton Code
    # inside working

    #  Class for block
    class Block:
        # constructor containing definition of parameters (sender name, transa
ction amount, receiver name, message and chain)
        def __init__(self, sender, transaction, receiver, message = ""):
           self.sender = sender
           self.transaction = transaction
           self.receiver = receiver
           self.message = message
           self.chain = []
        # function to create a block of transaction
        def create_block(self, p_hash = 0):
            b = { 'index': len(self.chain) + 1,                 #block number,
 for genesis block index = 0
                  'timestamp': str(datetime.datetime.now()),    #time of gener
ation of block
                  'transaction': self.transaction,              #transaction d
ata
                  'p_hash': p_hash,                             #hash of previ
ous block, for genesis block, p_hash = 0
                  'hash': self.generate_hash(p_hash)            #hash of curre
nt block
            }
            #append block to the chain to create a link
            self.chain.append(b)
            return b

        #access previous block of the chain
        def get_prev_block(self):
            return self.chain[-1]

        # function to generate hash using SHA256, hash algorithm created using
 transaction, sender, receiver data
        def generate_hash(self, p_hash):
            hash = hashlib.sha256((self.sender + str(self.transaction) + self.
receiver + self.message + str(p_hash)).encode()).hexdigest()
            return hash


    # function to retrieve the chain formed after transaction and mining of bl
ock
    #datatype dictionary, contains the chain of block and the length of chain
    def get_chain():
        response = { 'chain': S1.chain,
                     'length': len(S1.chain)}
        return response

    # mining block, generating chain of blocks linked via previous hash dataty
pe dictionary, contains the next block with following vaues
    #asigning hash of prev block to p_hash of current block for linking
    def mine_block():
        p_block = S1.get_prev_block()
        p_hash = p_block['hash']
        block = S1.create_block(p_hash)
        response = { 'index': block['index'],
                     'timestamp': block['timestamp'],
                     'p_hash':block['p_hash'],
                     'hash': block['hash']
```

```python
                     }

        return response

    #assigning form values entered by user to following variables
    Sender = request.form['one']
    Receiver = request.form['two']
    Coins = request.form['three']
    transaction = request.form['four']

    #accessing cursor of MySQL and firing query to write transaction details in table send of db login
    c = db.connection.cursor(MySQLdb.cursors.DictCursor)
    c.execute("INSERT INTO login.send(sender, receiver, transaction)VALUES(%s,%s,%s)", (Sender, Receiver, transaction))
    db.connection.commit()


    #assigning above values to new temporary variables
    #accessing date of creation for generating receipt
    S = Sender
    R = Receiver
    T = Coins
    time = datetime.datetime.now()

    #creating object S1 of class Block
    S1 = Block(S, T, R)

    #accessing function of class Block
    S1.create_block()

    #mining block 5 times
    for i in range(0,5):
        mine_block()

    C = get_chain()
    print(C)

    #END OF BLOCKCHAIN CODE

    #passing sender name, reciever id, transaction amount and time of generation
    #passing values to passdata.html file for receipt generation
    return render_template("passdata.html", s=Sender, r=Receiver, t=Coins, T=time)


@app.route('/profile', methods = ['GET', 'POST'])
def profile():
    return render_template("profile.html")

@app.route('/logout')
def out():
    return render_template("login.html")



if __name__ == '__main__':
    app.run()
```