```
from flask import Flask, render_template, url_for, redirect, request, flash, session

from flask_sqlalchemy import SQLAlchemy

from flask_login import UserMixin, login_user, LoginManager, login_required, logout_user

from flask_bcrypt import Bcrypt

from flask_cors import CORS

import requests



# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.

API_KEY = "hMlTgdcKCmEIQFtugTulqjmTZOcwxD7rXdL1HWiW9zRJ"

token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey": API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})

mltoken = token_response.json()["access_token"]


header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}



app = Flask(__name__)

CORS(app)

bcrypt = Bcrypt(app)

app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql://sql12564552:nKk1fhv756@sql12.freemysqlhosting.net/sql12564552'

app.config['SECRET_KEY'] = 'thisisasecretkey'

db = SQLAlchemy(app)

app.app_context().push()
```

```
predict = None


login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'



@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))



class User(db.Model, UserMixin):
    __tablename__ = 'users'
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(20), nullable=False, unique=True)
    password = db.Column(db.String(80), nullable=False)
    email = db.Column(db.String(80), nullable=False)

class Prediction(db.Model):
    __tablename__ = 'predict'
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(20), nullable=False)
    timestamp = db.Column(db.String(20), nullable=False)
    age = db.Column(db.Float, nullable=False)
```

```
bp = db.Column(db.Float, nullable=False)
sg = db.Column(db.Float, nullable=False)
al= db.Column(db.Float, nullable=False)
su= db.Column(db.Float, nullable=False)
rbc= db.Column(db.String(20), nullable=False)
pc= db.Column(db.String(20), nullable=False)
pcc= db.Column(db.String(20), nullable=False)
ba= db.Column(db.String(20), nullable=False)
bgr= db.Column(db.Float, nullable=False)
bu= db.Column(db.Float, nullable=False)
sc= db.Column(db.Float, nullable=False)
sod= db.Column(db.Float, nullable=False)
pot= db.Column(db.Float, nullable=False)
hemo= db.Column(db.Float, nullable=False)
pcv= db.Column(db.Float, nullable=False)
wc= db.Column(db.Float, nullable=False)
rc= db.Column(db.Float, nullable=False)
htn= db.Column(db.String(20), nullable=False)
dm= db.Column(db.String(20), nullable=False)
cad= db.Column(db.String(20), nullable=False)
appet= db.Column(db.String(20), nullable=False)
pe= db.Column(db.String(20), nullable=False)
ane= db.Column(db.String(20), nullable=False)
result = db.Column(db.String(20), nullable=False)
```

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



@app.route('/login', methods=['GET', 'POST'])
def login():
   if request.method == "POST":
      user = User.query.filter_by(username=request.form.get('username')).first()
      if user:
         if bcrypt.check_password_hash(user.password, request.form.get('password')):
            login_user(user)
            session['username'] = request.form.get('username')
            return redirect(url_for('dashboard'))
         else:
            flash("Password is Incorrect!")
   return render_template('login.html')

@app.route('/result', methods=['GET','POST'])
@login_required
def result():
   username = session['username']
   table = Prediction.query.filter_by(username=username).order_by(Prediction.timestamp.desc())
   return render_template('result.html',predict=predict, table=table)
```

```
@app.route('/dashboard', methods=['GET','POST'])
@login_required
def dashboard():
    form = request.form
    if request.method == "POST":
        age = float(form.get('age'))
        bp = float(form.get('bp'))
        sg = float(form.get('sg'))
        al= float(form.get('al'))
        su= float(form.get('su'))
        rbc= 0 if form.get('rbc') == 'normal' else 1
        pc= 0 if form.get('pc') == 'normal' else 1
        pcc= 0 if form.get('pcc') == 'notpresent' else 1
        ba= 0 if form.get('ba') == 'notpresent' else 1
        bgr= float(form.get('bgr'))
        bu= float(form.get('bu'))
        sc= float(form.get('sc'))
        sod= float(form.get('sod'))
        pot= float(form.get('pot'))
        hemo= float(form.get('hemo'))
        pcv= float(form.get('pcv'))
        wc= float(form.get('wc'))
        rc= float(form.get('rc'))
        htn= 0 if form.get('htn') == 'no' else 1
```

```
dm= 0 if form.get('dm') == 'no' else 1

cad= 0 if form.get('cad') == 'no' else 1

appet= 0 if form.get('appet') == 'good' else 1

pe= 0 if form.get('pe') == 'no' else 1

ane= 0 if form.get('ane') == 'no' else 1


print(age, bp, sg, al, su, rbc, pc, pcc, ba, bgr, bu, sc, sod, pot, hemo, pcv, wc, rc, htn, dm, cad, appet, pe, ane)

X = [[age, bp, sg, al, su, rbc, pc, pcc, ba, bgr, bu,sc, sod, pot, hemo, pcv, wc, rc, htn, dm, cad, appet, pe, ane]]

# NOTE: manually define and pass the array(s) of values to be scored in the next line

payload_scoring = {"input_data": [{"field": [['age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr', 'bu','sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad','appet', 'pe', 'ane']], "values": X}]}


response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/4bb20f2e-e060-412e-875e-a336e199f1aa/predictions?version=2022-11-11', json=payload_scoring, headers={'Authorization': 'Bearer ' + mltoken})

print(response_scoring)

predictions = response_scoring.json()

predict = predictions['predictions'][0]['values'][0][0]

print("Final prediction :",predict)

res = 'Positive' if predict==1 else 'Negative'

new_user = Prediction(

   username=form.get('username'),

   timestamp = form.get('timestamp'),

   age = age,

   bp = bp,
```

```
    sg = sg,
    al= al,
    su= su,
    rbc= form.get('rbc'),
    pc= form.get('pc'),
    pcc= form.get('pcc'),
    ba= form.get('ba'),
    bgr= bgr,
    bu= bu,
    sc= sc,
    sod= sod,
    pot= pot,
    hemo= hemo,
    pcv= pcv,
    wc= wc,
    rc= rc,
    htn= form.get('htn'),
    dm= form.get('dm'),
    cad= form.get('cad'),
    appet= form.get('appet'),
    pe= form.get('pe'),
    ane= form.get('ane'),
    result = res
)
db.session.add(new_user)
```

```
        db.session.commit()
        username = session['username']
        table = Prediction.query.filter_by(username=username).order_by(Prediction.timestamp.desc())
        print(table)
        return render_template('result.html', predict=predict, table=table)

    return render_template('dashboard.html')



@app.route('/logout', methods=['GET', 'POST'])
@login_required
def logout():
    logout_user()
    return redirect(url_for('login'))



@ app.route('/register', methods=['GET', 'POST'])
def register():
    form = request.form
    if request.method == "POST":
        existing_user = User.query.filter_by(username=form.get('username')).first()
        if existing_user:
            flash('That username already exists! Please choose a different one.')
        elif form.get('password') != form.get('cpassword'):
            flash('The password confirmation does not match!')
```

```
        else:
            hashed_password = bcrypt.generate_password_hash(form.get('password'))
            new_user = User(email=form.get('email'), username=form.get('username'),
password=hashed_password)
            db.session.add(new_user)
            db.session.commit()
            return redirect(url_for('login'))

    return render_template('register.html')



if __name__ == "__main__":
    app.run(debug=True)
```