**Assignment -1**

Python Programming

|  |  |
| --- | --- |
| Assignment Date | 10 September 2022 |
| Student Name | Yugesh V S |
| Student Roll Number | 7179KCTKCTKCTKCTKCTKCTKCT19BEC038 |
| Maximum Marks | 2 Marks |

**Question-1:**

1. Create registration page in html with username, email and phone number and by using POST method display it in next html page.

|  |
| --- |
| **Solution:** |
|  | Post method in action:  html form:  <html>  <head><title>POST Method in Action</title></head>  <body>  <form action="post.php" method="post">  Name <input type="text" name="user"><br />  Company<input type="text" name="comp"><br />  <input type="submit" value=" Submit Info">  </form>  </body>  </html>  <?php    $name = $\_POST['user'];  $company = $\_POST['comp'];    echo "My name is $name <br /> And I'm the CEO of my company {$company}";    ?> |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Question-2:**

2. Develop a flask program which should contain at least 5 packages used from pypi.org.

|  |  |  |
| --- | --- | --- |
| **Solution:**     |  |  | | --- | --- | |  | from flask import Flask, render\_template, url\_for, redirect  from flask\_sqlalchemy import SQLAlchemy  from flask\_login import UserMixin, login\_user, LoginManager, login\_required, logout\_user, current\_user  from flask\_wtf import FlaskForm  from wtforms import StringField, PasswordField, SubmitField  from wtforms.validators import InputRequired, Length, ValidationError  from flask\_bcrypt import Bcrypt  app = Flask(\_\_name\_\_)  db = SQLAlchemy(app)  bcrypt = Bcrypt(app)  app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///database.db'  app.config['SECRET\_KEY'] = 'thisisasecretkey'  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):  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)  class RegisterForm(FlaskForm):  username = StringField(validators=[  InputRequired(), Length(min=4, max=20)], render\_kw={"placeholder": "Username"})  password = PasswordField(validators=[  InputRequired(), Length(min=8, max=20)], render\_kw={"placeholder": "Password"})  submit = SubmitField('Register')  def validate\_username(self, username):  existing\_user\_username = User.query.filter\_by(  username=username.data).first()  if existing\_user\_username:  raise ValidationError(  'That username already exists. Please choose a different one.')  class LoginForm(FlaskForm):  username = StringField(validators=[  InputRequired(), Length(min=4, max=20)], render\_kw={"placeholder": "Username"})  password = PasswordField(validators=[  InputRequired(), Length(min=8, max=20)], render\_kw={"placeholder": "Password"})  submit = SubmitField('Login')  @app.route('/')  def home():  return render\_template('home.html')  @app.route('/login', methods=['GET', 'POST'])  def login():  form = LoginForm()  if form.validate\_on\_submit():  user = User.query.filter\_by(username=form.username.data).first()  if user:  if bcrypt.check\_password\_hash(user.password, form.password.data):  login\_user(user)  return redirect(url\_for('dashboard'))  return render\_template('login.html', form=form)  @app.route('/dashboard', methods=['GET', 'POST'])  @login\_required  def dashboard():  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 = RegisterForm()  if form.validate\_on\_submit():  hashed\_password = bcrypt.generate\_password\_hash(form.password.data)  new\_user = User(username=form.username.data, password=hashed\_password)  db.session.add(new\_user)  db.session.commit()  return redirect(url\_for('login'))  return render\_template('register.html', form=form)  if \_\_name\_\_ == "\_\_main\_\_":  app.run(debug=True) | |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |