### **Project 9: Web Application**

#### **Objective:**

Develop a simple web application using Flask and integrate it with a front-end framework like Bootstrap. The app will allow users to interact with a database (e.g., a user registration and login system).

### **Instructions**

#### **Step 1: Set Up the Environment**

1. Create a new Python file called web\_app.py.

Install the required libraries:  
Copy code  
pip install flask flask\_sqlalchemy flask\_bcrypt flask\_login

1. Create a folder called templates for HTML files and a folder called static for CSS and JavaScript files.

#### **Step 2: Create the Flask App**

Import the necessary libraries and initialize the Flask app:  
python  
Copy code  
from flask import Flask, render\_template, redirect, url\_for, request, flash

from flask\_sqlalchemy import SQLAlchemy

from flask\_bcrypt import Bcrypt

from flask\_login import LoginManager, UserMixin, login\_user, current\_user, logout\_user, login\_required

app = Flask(\_\_name\_\_)

app.config['SECRET\_KEY'] = 'your\_secret\_key'

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///site.db'

db = SQLAlchemy(app)

bcrypt = Bcrypt(app)

login\_manager = LoginManager(app)

login\_manager.login\_view = 'login'

login\_manager.login\_message\_category = 'info'

#### **Step 3: Create Database Models**

Define a User model with fields for id, username, email, and password:  
python  
Copy code  
class User(db.Model, UserMixin):

id = db.Column(db.Integer, primary\_key=True)

username = db.Column(db.String(20), unique=True, nullable=False)

email = db.Column(db.String(120), unique=True, nullable=False)

password = db.Column(db.String(60), nullable=False)

def \_\_repr\_\_(self):

return f"User('{self.username}', '{self.email}')"

Create the database using the following commands:  
python  
Copy code  
from web\_app import db

db.create\_all()

#### **Step 4: Create User Registration and Login Forms**

Create a file called forms.py and define forms using Flask-WTF:  
python  
Copy code  
from flask\_wtf import FlaskForm

from wtforms import StringField, PasswordField, SubmitField, BooleanField

from wtforms.validators import DataRequired, Length, Email, EqualTo, ValidationError

from web\_app.models import User

class RegistrationForm(FlaskForm):

username = StringField('Username', validators=[DataRequired(), Length(min=2, max=20)])

email = StringField('Email', validators=[DataRequired(), Email()])

password = PasswordField('Password', validators=[DataRequired()])

confirm\_password = PasswordField('Confirm Password', validators=[DataRequired(), EqualTo('password')])

submit = SubmitField('Sign Up')

def validate\_username(self, username):

user = User.query.filter\_by(username=username.data).first()

if user:

raise ValidationError('Username is already taken.')

def validate\_email(self, email):

user = User.query.filter\_by(email=email.data).first()

if user:

raise ValidationError('Email is already registered.')

class LoginForm(FlaskForm):

email = StringField('Email', validators=[DataRequired(), Email()])

password = PasswordField('Password', validators=[DataRequired()])

remember = BooleanField('Remember Me')

submit = SubmitField('Login')

#### **Step 5: Create Registration and Login Routes**

Create routes for user registration and login:  
python  
Copy code  
@app.route("/register", methods=['GET', 'POST'])

def register():

if current\_user.is\_authenticated:

return redirect(url\_for('home'))

form = RegistrationForm()

if form.validate\_on\_submit():

hashed\_password = bcrypt.generate\_password\_hash(form.password.data).decode('utf-8')

user = User(username=form.username.data, email=form.email.data, password=hashed\_password)

db.session.add(user)

db.session.commit()

flash('Your account has been created! You can now log in.', 'success')

return redirect(url\_for('login'))

return render\_template('register.html', title='Register', form=form)

@app.route("/login", methods=['GET', 'POST'])

def login():

if current\_user.is\_authenticated:

return redirect(url\_for('home'))

form = LoginForm()

if form.validate\_on\_submit():

user = User.query.filter\_by(email=form.email.data).first()

if user and bcrypt.check\_password\_hash(user.password, form.password.data):

login\_user(user, remember=form.remember.data)

next\_page = request.args.get('next')

return redirect(next\_page) if next\_page else redirect(url\_for('home'))

else:

flash('Login Unsuccessful. Please check email and password.', 'danger')

return render\_template('login.html', title='Login', form=form)

@app.route("/logout")

def logout():

logout\_user()

return redirect(url\_for('home'))

#### **Step 6: Create HTML Templates**

1. Create register.html and login.html in the templates folder with forms to collect user input.
2. Create a base.html file for the layout and include a navbar with links to Home, Register, and Login.

#### **Step 7: Create a Home Route and Restrict Access**

Create a home route and protect it using the @login\_required decorator:  
python  
Copy code  
@app.route("/home")

@login\_required

def home():

return render\_template('home.html', title='Home')

#### **Step 8: Run the Application**

Run the application and test the registration and login functionality:  
python  
Copy code  
if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

#### **Step 9: Test the Application**

1. Register a new user and ensure the user is saved in the database.
2. Log in with the registered user and confirm access to the home page.