03 – Forms and User Input

အချက်အလက်တွေကို server မှ user တစ်ဖက်တည်း စီးဆင်းခွင့် ပေးမယ်ဆိုရင် unidirectional ဖြစ်ပြီး၊ applications အများစုမှာတော့ အချက်အလက်က အပြန်အလှန် စီးဆင်းဖို့ လိုအပ်ပါတယ်။ user မှ ပို့လွှတ်လိုက်တဲ့ အချက်အလက်ကို server က လက်ခံပြီး processes လုပ်နိုင်ရပါမယ်။

အဲဒီအတွက် HTML ရဲ့ form element နဲ့ web forms ဖန်တီးပြီး အချက်အလက်ကို user က ရိုက်ထည့်ပေးနိုင်ပါတယ်။ form ရဲ့ POST request ကိုသုံးပြီး web browser မှ server ထံ form data ကို submitted လုပ်ပေးနိုင်ပါတယ်။

ဒါပေမယ့် ရိုးရိုး HTML form အစား Flask framework မှာတော့ Flask–WTF extension ကို သုံးပြီး စိတ်တိုင်းကျ web forms တွေနဲ့ အလုပ်လုပ်ပါမယ်။ WTF ဆိုတာ WT Forms ကို ဆိုလိုပြီး၊ အသုံးပြုသူတွေအတွက် အပြန်အလှန် အကျိုးသက်ရောက်နိုင်တဲ့ user interface များ ထောက်ပံ့ဖို့ ရည်ရွယ်ပါတယ်။ WTF က flask ရဲ့ built–in module တစ်ခုဖြစ်ပြီး၊ flask web applications များအတွက် နည်းလမ်းအသွယ်သွယ်နဲ့ forms များကို ထောက်ပံ့ပေးပါတယ်။ CSRF token နှင့်အတူ Global CSRF protection. reCAPTCHA support ပေးတဲ့ Secure Form တစ်ခုပါ။

WTForms ကို သုံးဖို့ ပထမဆုံး Flask–WTF ကို install လုပ်ပါမယ်။

> Terminal ကို သွားပါ။ virtualenv ကို activate လုပ်ပါ။
>
> (venv) pip install flask–wtf

Flask–WTF မှ FlaskForm class ကို inherits လုပ်ပြီး class တစ်ခုချင်းစီဖန်တီးပေးခြင်းအားဖြင့် web form တွေကို server မှာ တင်ပြနိုင်ပါတယ်။ class ထဲမှာတော့ form မှာ လိုအပ်တဲ့ list of fields ကို object တစ်ခုစီအနေနဲ့ define လုပ်ပေးရပါမယ်။ ယင်း object field တစ်ခုစီမှာလဲ တစ်ခု (သို့) တစ်ခုမကသော validators များပါရှိပါမယ်။ validator ကတော့ function တစ်ခုဖြစ်ပြီး user မှ ပို့လွှတ်သော data များ valid ဖြစ်မဖြစ်ကို check လုပ်ပေးရပါတယ်။

ဒါဆိုရင် FlaskForm class ကို extend လုပ်ပြီး၊ Register နဲ့ Login လုပ်ဖို့ RegistrationForm နဲ့ LoginForm class တွေကို forms.py file မှာ သွားရေးပါမယ်။ FlaskForm class ကို flask_wtf module မှ import လုပ်ရပါမယ်။

RegistrationForm မှာ username, email, password, confirm_password နဲ့ submit button ဆိုတဲ့ fields (class variables) တွေပါပါမယ်။ username နဲ့ email က StringField ဖြစ်ပြီး၊ password တွေအတွက် PasswordField ၊ နောက် submit အတွက် SubmitField တွေကို သုံးရပါမယ်။ ဒါကြောင့် သက်ဆိုင်ရာ function တွေကို wtforms ကနေ import လုပ်ထားရပါမယ်။ နောက်တစ်ခုက သက်ဆိုင်ရာ field function တွေမှာ ပထမဆုံး argument အနေနဲ့ label ပေးရပြီး၊ နောက် validators

တွေကို list တစ်ခုနဲ့ ထည့်ပေးရပါမယ်။ အဲဒီအတွက်လည်း လိုတာတွေကို wtforms.validators မှ import လုပ်ထားဖို့ လိုအပ်ပါတယ်။ forms.py file ကို ကြည့်ပါ။

forms.py

```
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField, BooleanField
from wtforms.validators import DataRequired, Length, Email, EqualTo

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')

class LoginForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[DataRequired()])
    remember = BooleanField('Remember Me')
    submit = SubmitField('Login')
```

Form ဖန်တီးပြီးရင်တော့ register နဲ့ login အတွက် route ကို ရေးပါမယ်။ flaskblog.py ကိုဖွင့်ပါ။ ပထမ forms.py မှ RegistrationForm နဲ့ LoginForm ကို import လုပ်ပါ။ WTForm က secure form ဆိုတာသိခဲ့ပါတယ်။ ဒါကြောင့် secret key ကို configure လုပ်ပေးဖို့ လိုအပ်ပါတယ်။ secret key ဆိုတာက random နဲ့ unique ဖြစ်တဲ့ content တွေပါတဲ့ string တစ်ခုပါ။ နည်းလမ်းအသွယ်သွယ်နဲ့ applications တွေရဲ့ လုံခြုံရေး တိုးမြှင့်ဖို့ encryption or singing key အနေနဲ့ သုံးပါတယ်။ secret key ကို configure လုပ်ပေးတဲ့အတွက် cross-site request forgery (CSRF) attacks မှ

ကာကွယ်ပေးပါတယ်။ CSRF attack က မိမိလက်ရှိ login ဝင်ပြီး အသုံးပြုနေတဲ့ application server ကို malicious website က request ပို့လွှတ်တဲ့အချိန်မှာ ဖြစ်ပေါ်တတ်ပါတယ်။

secret key ကို create လုပ်ပါမယ်။

Terminal မှ Python shell ကို သွားပါ။

```
>>> import secrets
>>> secrets.token_hex(16)
'5791628bb0b13ce0c676dfde280ba245'
```

ရလာတဲ့ secret key ကို copy ကူးပြီး flaskblog.py မှာ configure လုပ်ရပါမယ်။

app.route decorator နဲ့ register / login route ကို ရေးပါမယ်။

```
@app.route("/register")
def register():
        form = RegistrationForm()
    return render_template('register.html', title='Register', form=form)


@app.route("/login")
def login():
        form = LoginForm()
    return render_template('login.html', title='login', form=form)
```

Render လုပ်ဖို့ register.html နဲ့ login.html files တွေကို templates folder အောက်မှာ ဖန်တီးပေးရပါမယ်။ register form ကို content-section မှာ ဖော်ပြမှာဖြစ်ပြီး <form> tag ကို method [POST] နဲ့ ဖန်တီးပါမယ်။ csrf token စစ်ဖို့ form element တစ်ခုဖြစ်တဲ့ form.hidden_tag() ကိုထည့်ပါမယ်။ ဒီ element က hidden ဖြစ်နေတဲ့ form field တစ်ခုဖြစ်ပြီး CSRF protection ကို အကောင်အထည်ဖော်ဖို့ Flask-WTF ကနေ အသုံးပြုပါမယ်။ နောက် register လုပ်ဖို့ field (4) ခုကို Join Today legend နဲ့ အုပ်စုဖွဲ့ပြီး ထည့်ထားပါမယ်။ အဲဒီနောက်မှာ submit button ရယ် Already Have An Account? စာသားနဲ့ Sign In link ကို text-muted လုပ်ပြီး ရေးထားပါမယ်။

register.html file ကို ကြည့်ပါ။

register.html

```
{% extends "layout.html" %}
{% block content %}
    <div class="content-section">
        <form method="POST" action="">
            {{ form.hidden_tag() }}
            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Join Today</legend>
                <div class="form-group">
                    {{ form.username.label(class="form-control-label") }}
                    {{ form.username(class="form-control form-control-lg") }}
                </div>
                <div class="form-group">
                    {{ form.email.label(class="form-control-label") }}
                    {{ form.email(class="form-control form-control-lg") }}
                </div>
                <div class="form-group">
                    {{ form.password.label(class="form-control-label") }}
                    {{ form.password(class="form-control form-control-lg") }}
                </div>
                <div class="form-group">
                    {{ form.confirm_password.label(class="form-control-label") }}
                    {{ form.confirm_password(class="form-control form-control-lg") }}
                </div>
            </fieldset>


            <div class="form-group">
                {{ form.submit(class="btn btn-outline-info") }}
            </div>
        </form>
```

```
    </div>
    <div class="border-top pt-3">
        <small class="text-muted">
            Already Have An Account? <a class="ml-2" href="{{ url_for('login') }}">Sign In</a>
        </small>
    </div>
{% endblock content %}
```

file ကို run ကြည့်လိုက်ပါ။ Not allowed Method ဆိုတဲ့ errors ကို browser မှာ ပြပေးပါလိမ့်မယ်။

အဲဒါကတော့ register route မှာ methods argument ကို ထည့်မပေးခဲ့လို့ပါ။ app.route decorator မှာ method argument ကို ထည့်ပြီး URL map မှာ GET နဲ့ POST အတွက် view function က handler တစ်ခုအနေနဲ့ register လုပ်ဖို့ Flask ကို ပြောပေးရမှာပါ။ အဲလိုပြောမထားရင် view function က GET request ကိုသာ handle လုပ်ပါတယ်။ အခု form data ကို POST request နဲ့ လွှတ်တာဖြစ်တဲ့အတွက် methods argument ထည့်ပေးရပါမယ်။

file ကို ပြန် run ကြည့်တဲ့အခါ method error အဆင်ပြေသွားပေမဲ့ form field တွေမှာ ထည့်ထားတဲ့ validators တွေစစ်ဖို့ ကျန်နေပါတယ်။ ဥပမာ – username ပါမပါ၊ နောက် သတ်မှတ်ထားတဲ့ min, max length ကိုက်မကိုက် အစရှိသည်တို့ စစ်ဖို့လိုပါတယ်။ field validators အားလုံးအနေအားဖြင့် data ကို လက်ခံပြီး submit လုပ်လိုက်တဲ့အခါ form ရဲ့ valiate_on_submit() method က True or False ကို return ပြန်ပေးပါတယ်။ အဲဒီ return value အရ form က render လုပ်ရမလား၊ ဒါမှမဟုတ် အခြား process လုပ်ပေးရမလား ဆုံးဖြတ်ရပါတယ်။ ဒါကြောင့် အဲဒီအတွက် လိုအပ်သလို code ကိုပြင်ရေးရပါမယ်။

နောက် request က အပြီးသတ်လုပ်ဆောင်ချက် status update ကို message ပေးဖို့လည်း လိုပါတယ်။ confirmation လား၊ warning လား၊ error လားဆိုတာ message နဲ့ ဖော်ပြဖို့ flash() function ကို သုံးပါမယ်။ flash message ကို ဖော်ပြဖို့အတွက် template က message ကို render လုပ်ပေးဖို့လိုပါတယ်။ layout.html ဆိုတဲ့ base template file မှာ control block နဲ့ ဖမ်းပေးရပါမယ်။ flask message ကို bootstrap နဲ့ သုံးမယ်ဆိုရင် category ထည့်ပေးရပါမယ်။ route ကို ပြင်ပေးရမှာပါ။

register အတွက် အားလုံး အဆင်ပြေသွားရင် login အတွက်လည်း အလွယ်တကူ ရေးလိုက်လို့ ရပါပြီ။ login.html file ကို ကြည့်ပါ။ login form မှာ email ကို admin@blog.com လို့ပေးပြီး password ကိုလည်း password လို့ပဲ ပေးရပါမယ်။ ဒါမှ login လုပ်လို့ရပါလိမ့်မယ်။ အဲလိုမှမဟုတ်ရင် message ပြဖို့ပြောထားတာပါ။ file အားလုံးရေးပြီးရင် run ကြည့်ပါမယ်။

flaskblog.py

```
from flask import Flask, render_template, url_for, flash, redirect
from forms import RegistrationForm, LoginForm
app = Flask(__name__)
app.config['SECRET_KEY'] = '5791628bb0b13ce0c676dfde280ba245'
posts = [
    {
        ...
     },
    {
        ...
    }
]

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

@app.route("/about")
def about():
    return render_template('about.html', title='About')
@app.route("/register", methods=['GET', 'POST'])
def register():
    form = RegistrationForm()
    if form.validate_on_submit():
        flash(f'Account created for {form.username.data}!', 'success')
        return redirect(url_for('home'))
    return render_template('register.html', title='Register', form=form)
```

```
@app.route("/login", methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        if form.email.data == 'admin@blog.com' and form.password.data == 'password':
            flash('You have been logged in!', 'success')
            return redirect(url_for('home'))
        else:
            flash('Login Unsuccessful. Please check username and password', 'danger')
    return render_template('login.html', title='Login', form=form)

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

forms.py

```
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField, BooleanField
from wtforms.validators import DataRequired, Length, Email, EqualTo

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')

class LoginForm(FlaskForm):
        email = StringField('Email', validators=[DataRequired(), Email()])
        password = PasswordField('Password', validators=[DataRequired()])
        remember = BooleanField('Remember Me')
        submit = SubmitField('Login')
```

register.html

```
{% extends "layout.html" %}
{% block content %}
    <div class="content-section">
        <form method="POST" action="">
            {{ form.hidden_tag() }}
            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Join Today</legend>
                <div class="form-group">
                    {{ form.username.label(class="form-control-label") }}
                    {% if form.username.errors %}
                        {{ form.username(class="form-control form-control-lg is-invalid") }}
                        <div class="invalid-feedback">
                            {% for error in form.username.errors %}
                                <span>{{ error }}</span>
                            {% endfor %}
                        </div>
                    {% else %}
                        {{ form.username(class="form-control form-control-lg") }}
                    {% endif %}
                </div>
                 <div class="form-group">
                    {{ form.email.label(class="form-control-label") }}
                    {% if form.email.errors %}
                        {{ form.email(class="form-control form-control-lg is-invalid") }}
                        <div class="invalid-feedback">
                            {% for error in form.email.errors %}
                                <span>{{ error }}</span>
                            {% endfor %}
```

```
        </div>
    {% else %}
        {{ form.email(class="form-control form-control-lg") }}
    {% endif %}
</div>
<div class="form-group">
    {{ form.password.label(class="form-control-label") }}
    {% if form.password.errors %}
        {{ form.password(class="form-control form-control-lg is-invalid") }}
        <div class="invalid-feedback">
            {% for error in form.password.errors %}
                <span>{{ error }}</span>
            {% endfor %}
        </div>
    {% else %}
        {{ form.password(class="form-control form-control-lg") }}
    {% endif %}
</div>
<div class="form-group">
    {{ form.confirm_password.label(class="form-control-label") }}
    {% if form.confirm_password.errors %}
            {{ form.confirm_password(class="form-control form-control-lg is-
            invalid") }}
                <div class="invalid-feedback">
                {% for error in form.confirm_password.errors %}
                <span>{{ error }}</span>
                {% endfor %}
                </div>
      {% else %}
      {{ form.confirm_password(class="form-control form-control-lg") }}
```

```
                {% endif %}
            </div>
            </fieldset>
            <div class="form-group">
                {{ form.submit(class="btn btn-outline-info") }}
            </div>
        </form>
    </div>
    <div class="border-top pt-3">
        <small class="text-muted">
            Already Have An Account? <a class="ml-2" href="{{ url_for('login') }}">Sign In</a>
        </small>
    </div>
{% endblock content %}
```

login.html

```
{% extends "layout.html" %}
{% block content %}
    <div class="content-section">
        <form method="POST" action="">
            {{ form.hidden_tag() }}
            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Log In</legend>
                <div class="form-group">
                    {{ form.email.label(class="form-control-label") }}
                    {% if form.email.errors %}
                        {{ form.email(class="form-control form-control-lg is-invalid") }}
                        <div class="invalid-feedback">
                            {% for error in form.email.errors %}
                                <span>{{ error }}</span>
                            {% endfor %}
                        </div>
                    {% else %}
                        {{ form.email(class="form-control form-control-lg") }}
                    {% endif %}
                </div>
                <div class="form-group">
```

```
            {{ form.password.label(class="form-control-label") }}
            {% if form.password.errors %}
                {{ form.password(class="form-control form-control-lg is-invalid") }}
                <div class="invalid-feedback">
                    {% for error in form.password.errors %}
                        <span>{{ error }}</span>
                    {% endfor %}
                </div>
            {% else %}
                {{ form.password(class="form-control form-control-lg") }}
            {% endif %}
        </div>
        <div class="form-check">
            {{ form.remember(class="form-check-input") }}
            {{ form.remember.label(class="form-check-label") }}
        </div>
    </fieldset>
    <div class="form-group">
        {{ form.submit(class="btn btn-outline-info") }}
    </div>
<small class="text-muted ml-2">
        <a href="#">Forgot Password?</a>
```

```
            </small>
        </form>
    </div>
    <div class="border-top pt-3">
        <small class="text-muted">
            Need An Account? <a class="ml-2" href="{{ url_for('register') }}">Sign Up Now</a>
        </small>
    </div>
{% endblock content %}
```

layout.html

```
<!DOCTYPE html>
<html>
<head>
 ...
</head>
<body>
    <header class="site-header">
      ...
    </header>
    <main role="main" class="container">
      <div class="row">
        <div class="col-md-8">
          {% with messages = get_flashed_messages(with_categories=true) %}
            {% if messages %}
              {% for category, message in messages %}
```

```
            <div class="alert alert-{{ category }}">
              {{ message }}
            </div>
          {% endfor %}
        {% endif %}
      {% endwith %}
      {% block content %}{% endblock %}
    </div>
    <div class="col-md-4">
      <div class="content-section"> ...
      </div>
    </div>
  </div>
</main>
<!-- Optional JavaScript -->
  ...
</body>
</html>
```