3.2. Student Handout

Flask Forms and Handling User Input: Student Handout

Introduction to Flask Forms

What is a Form?

- A form is a way for users to send data to a web application.
- In Flask, forms are created using HTML to collect data from users and send it to the server for processing.

Example HTML Form

```
<form action="/submit" method="POST">

<label for="name">Name:</label>

<input type="text" id="name" name="name">

<label for="email">Email:</label>

<input type="email" id="email" name="email">

<input type="submit" value="Submit">

</form>
```

Using HTML Forms to Send Data to Flask

How Does a Form Work?

- `<form>`: Defines the form with `action` (where data is sent) and `method` (how data is sent).
- `<input>`: Fields for user data entry, identified by the `name` attribute.
- `<submit>`: Button to submit the form.

Flask Route to Handle Form Submission

```
from flask import Flask, request


app = Flask(__name__)


@app.route('/submit', methods=['POST'])
def submit_form():
name = request.form['name']
email = request.form['email']
return f"Name: {name}, Email: {email}"
```

Handling GET and POST Requests

What are GET and POST Requests?

- **GET**: Data is appended to the URL, visible and less secure.
- **POST**: Data is sent in the request body, more secure.

Handling Both GET and POST in Flask

```
@app.route('/submit', methods=['GET', 'POST'])
def submit_form():
if request.method == 'POST':
name = request.form['name']
email = request.form['email']
return f"Name: {name}, Email: {email}"
```

```
else:

return "Please submit the form."
```

Form Data Processing on the Server Side

Processing Form Data

- Save data to a database.
- Send an email.
- Perform calculations.

Accessing Form Data in Flask

```
name = request.form['name']

email = request.form['email']
```

Validating Form Input with Flask-WTF

What is Flask-WTF?

- An extension for Flask that simplifies form handling and validation using WTForms.

Example Form with Validation

```
from flask_wtf import FlaskForm

from wtforms import StringField, EmailField, SubmitField

from wtforms.validators import DataRequired, Email



class MyForm(FlaskForm):

name = StringField('Name', validators=[DataRequired()])

email = EmailField('Email', validators=[DataRequired(), Email()])
```

```
submit = SubmitField('Submit')
```

Error Handling and Displaying Error Messages in Templates

Displaying Error Messages

```
<form method="POST">

{{ form.hidden_tag() }}

<div>

{{ form.name.label }}<br>

{{ form.name(size=20) }}<br>

{% if form.name.errors %}

<ul>

{% for error in form.name.errors %}

<li>{{ error }}</li>

{% endfor %}

</ul>

{% endif %}

</div>

<div>

{{ form.email.label }}<br>

{{ form.email(size=20) }}<br>

{% if form.email.errors %}
```

```
<ul>

{% for error in form.email.errors %}

<li>{{ error }}</li>

{% endfor %}

</ul>

{% endif %}

</div>

{{ form.submit() }}

</form>
```

Redirects and Message Flashing After Form Submission

Redirecting After Form Submission

```
from flask import redirect, url_for



@app.route('/submit', methods=['POST'])

def submit_form():

# Process the form data

return redirect(url_for('thank_you'))
```

Flashing Messages

```
from flask import flash
```

```
@app.route('/submit', methods=['POST'])

def submit_form():

# Process the form data

flash('Form submitted successfully!')

return redirect(url_for('thank_you'))
```

Displaying Flashed Messages

```
{% with messages = get_flashed_messages() %}

{% if messages %}

<ul>

{% for message in messages %}

<li>{{ message }}</li>

{% endfor %}

</ul>

{% endif %}

{% endwith %}
```

Conclusion

Key Steps in Handling Forms and User Input in Flask

1. **Creating HTML forms** to collect data.
2. **Handling GET and POST requests** to send data to the server.
3. **Processing form data** on the server side.
4. **Validating form input** using Flask-WTF.
5. **Handling errors** and displaying error messages.
6. **Redirecting and flashing messages** after form submission.

Diagram: Flow of Form Submission in Flask

```
User --> Fills Form --> Submits Form --> Flask Server --> Processes Data -->
Redirects/Flashes Message --> User
```

Thank you for your attention! If you have any questions, feel free to ask.