Experiment 1: Implement a simple demonstration of creating, retrieving, and deleting: 1) Cookies 2) Sessions

Objective:
Demonstrate Flask's client-side (cookies) and server-side (sessions) state management by creating a web app that:

- Stores/retrieves user data in cookies
- Manages user sessions with server-side storage
- Clears stored data on demand

1.1 Cookie Implementation

What we built:

- Single HTML page with:
  - Form to submit username (POST to `/setcookie`)
  - Link to check cookie (`/getcookie`)
  - Link to delete cookie (`/clearcookie`)
- **Displays:**
  - Confirmation when cookie is set/deleted
  - Personalized greeting when cookie exists
  - Error if no cookie found

```
from flask import Flask, request, make_response, render_template_string

app = Flask(__name__)

# HTML template with a form and links to get and clear cookies.
template = """
<!doctype html>
<title>Cookie Demo</title>
<h2>Cookie Example</h2>

<!-- Simple form to accept user's name -->
<form method="POST" action="/setcookie">
  Enter your name: <input type="text" name="username">
  <input type="submit" value="Set Cookie">
</form>
<br>
```

```
<!-- Links to check and clear cookies -->
<a href="/getcookie">Check Cookie</a> <br>
<a href="/clearcookie">Clear Cookie</a>
"""

@app.route('/')
def home():
    # render_template_string is used to render inline HTML content
    return render_template_string(template)

@app.route('/setcookie', methods=['POST'])
def setcookie():
    username = request.form.get('username')

    # make_response allows us to attach cookies to a response object
    resp = make_response("Cookie has been set for user: {}<br><a href='/'>Go back</a>".fo

    if username:
        # set_cookie sets a cookie key-value pair in the response
        resp.set_cookie('username', username)
    return resp

@app.route('/getcookie')
def getcookie():
    # request.cookies is used to access cookies sent by the browser
    username = request.cookies.get('username')

    if username:
        return f"Hello {username}, welcome back!"
    else:
        return "No cookie found. Please enter your name first."

@app.route('/clearcookie')
def clearcookie():
    # make_response allows modification of the response before returning
    resp = make_response("Cookie has been cleared!")

    # Setting cookie value to empty and expiry to 0 deletes the cookie
    resp.set_cookie('username', '', expires=0)
    return resp

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

Key Features:

- Sets cookies via `set_cookie()`

- Retrieves with `request.cookies`
- Deletes via expiration overwrite

1.2 Session Implementation

What we built:

- Identical HTML structure to cookie demo for comparison:
  - Form to submit username (POST to `/setsession`)
  - Link to check session (`/getsession`)
  - Link to clear session (`/clearsession`)
- **Displays:**
  - Session confirmation messages
  - Server-side stored username
  - Clear visual feedback on session state

```
from flask import Flask, session, request, render_template_string

app = Flask(__name__)
app.secret_key = 'your_secret_key_here'  # Required to use sessions securely

# HTML template with a form and links to get and clear session data
template = """
<!doctype html>
<title>Session Demo</title>
<h2>Session Example</h2>

<!-- Simple form to accept user's name -->
<form method="POST" action="/setsession">
  Enter your name: <input type="text" name="username">
  <input type="submit" value="Set Session">
</form>
<br>

<!-- Links to check and clear session -->
<a href="/getsession">Check Session</a> <br>
<a href="/clearsession">Clear Session</a>
"""

@app.route('/')
def home():
    # render_template_string is used to embed HTML directly in Python
    return render_template_string(template)
```

```
@app.route('/setsession', methods=['POST'])
def setsession():
    username = request.form.get('username')

    # Session data is stored server-side, but a session ID is saved in the user's browser
    session['username'] = username

    return "Session has been set for user: {}<br><a href='/'>Go back</a>".format(username

@app.route('/getsession')
def getsession():
    # Accessing session data using session dictionary
    username = session.get('username')

    if username:
        return f"Hello {username}, you are logged in via session!"
    else:
        return "No session found. Please enter your name first."

@app.route('/clearsession')
def clearsession():
    # session.pop removes 'username' key from session storage
    session.pop('username', None)
    return "Session has been cleared!<br><a href='/'>Go back</a>"

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

Key Features:

- Uses server-side `session` dictionary
- Requires secret key for security
- Clears data with `session.pop()`

Submission:

Working code with demonstration screenshots in records and written output in observations.