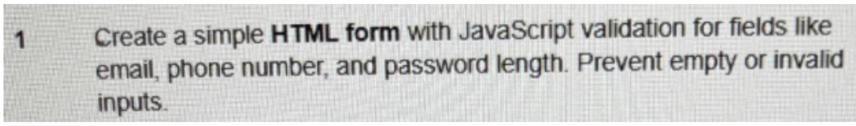
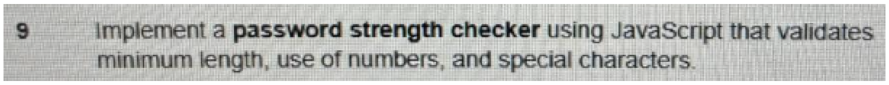


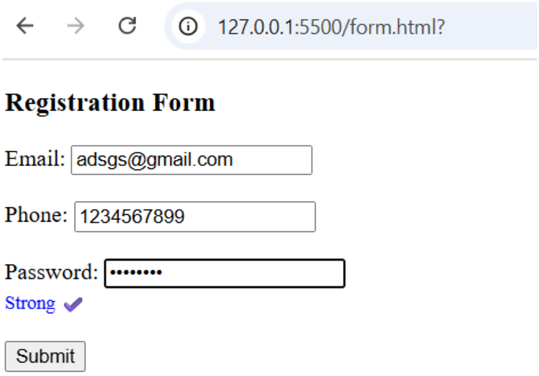
1 Create a simple **HTML form** with JavaScript validation for fields like email, phone number, and password length. Prevent empty or invalid inputs.



9 Implement a **password strength checker** using JavaScript that validates minimum length, use of numbers, and special characters.

```
<!DOCTYPE html>
<html>
<body>
<h3>Registration Form</h3>
<form onsubmit="return validate()">
  Email: <input type="text" id="email"><br><br>
  Phone: <input type="text" id="phone"><br><br>
  Password: <input type="password" id="pass" onkeyup="checkPass()"><br>
  <small id="msg" style="color:blue;"></small><br><br>
  <button type="submit">Submit</button>
</form>
<script>
function validate() {
  let e = document.getElementById("email").value;
  let p = document.getElementById("phone").value;
  let pass = document.getElementById("pass").value;
  if (e == "" || p == "" || pass == "") {
    alert("Fields cannot be empty");
    return false;
  }
  if (!e.includes("@") || !e.includes(".")) {
    alert("Invalid email");
    return false;
  }
  if (p.length != 10 || isNaN(p)) {
    alert("Phone must be 10 digits");
    return false;
  }
  if (pass.length < 8 || !/\d/.test(pass) || !/[!@#$%]/.test(pass)) {
    alert("Weak password");
    return false;
  }
  alert("Form Submitted!");
  return true;
}
```

```
function checkPass() {
  let pass = document.getElementById("pass").value;
  let msg = "Strong Password";
  if (pass.length < 8) msg = "Min 8 characters";
  else if (!/\d/.test(pass)) msg = "Add a number";
  else if (!/[!@#$%]/.test(pass)) msg = "Add a special char";
  document.getElementById("msg").innerHTML = msg;
}
</script>
</body>
</html>
```



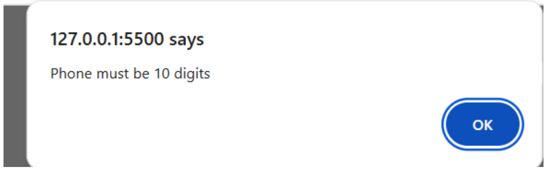
Registration Form

Email: adsgs@gmail.com

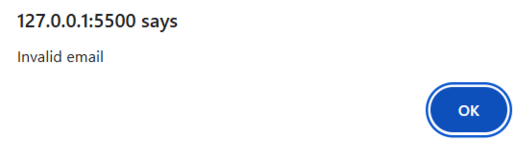
Phone: 1234567899

Password: ••••••••

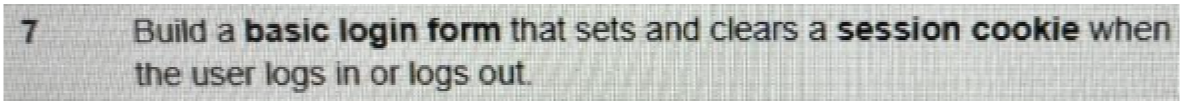
Strong ✓


127.0.0.1:5500 says

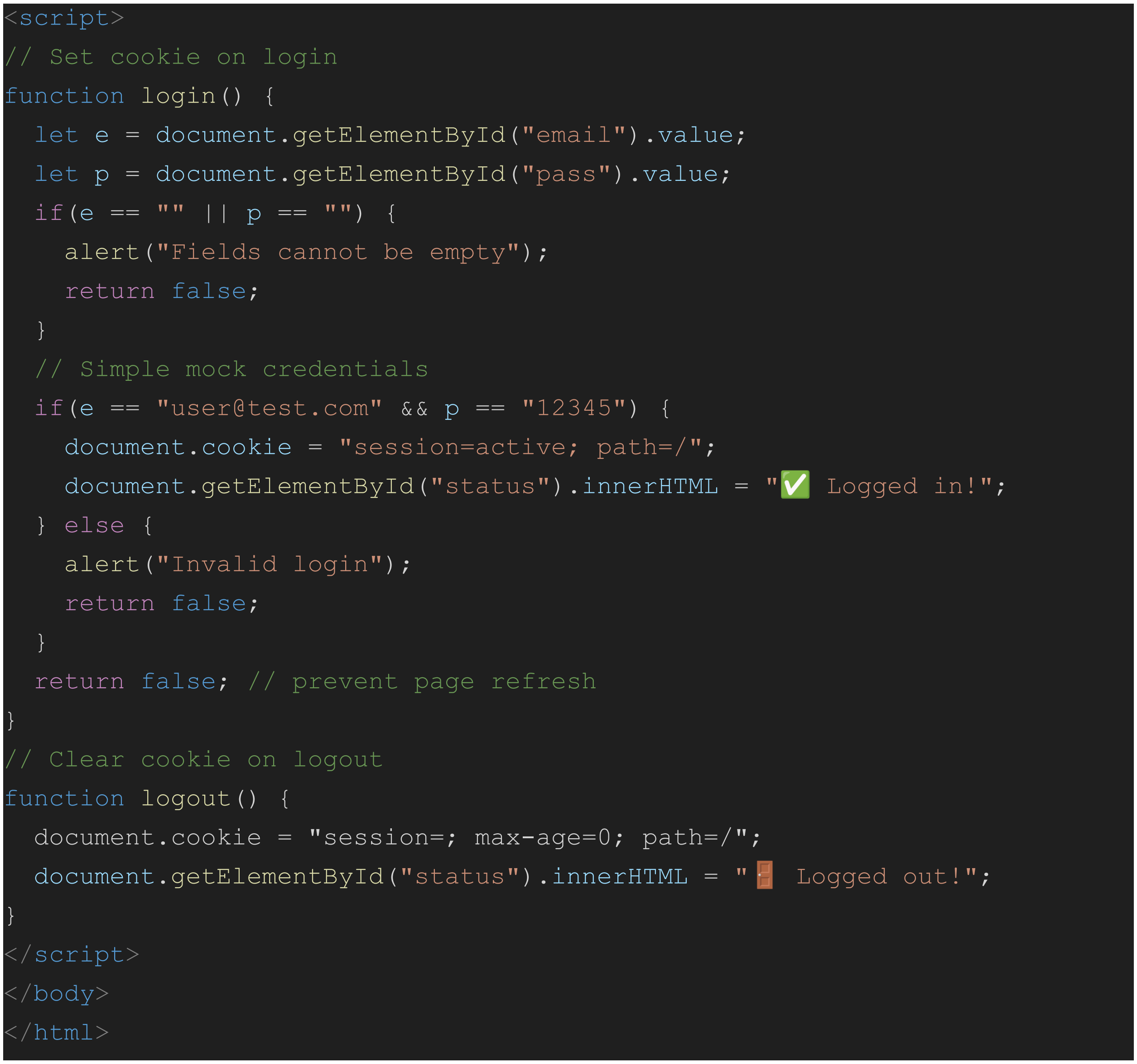
Phone must be 10 digits


127.0.0.1:5500 says

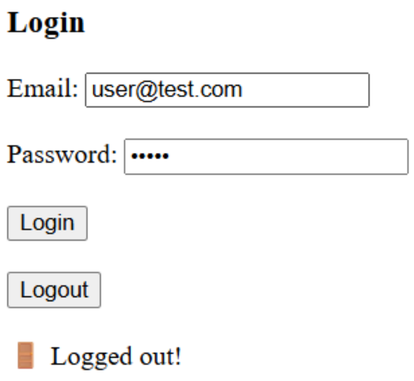
Invalid email


7 Build a **basic login form** that sets and clears a **session cookie** when the user logs in or logs out.

```
<!DOCTYPE html>
<html>
<body>
<h3>Login</h3>
<form onsubmit="return login()">
  Email: <input type="text" id="email"><br><br>
  Password: <input type="password" id="pass"><br><br>
  <button type="submit">Login</button>
</form>
<br>
<button onclick="logout()">Logout</button>
<p id="status"></p>
```



```
<script>
// Set cookie on login
function login() {
  let e = document.getElementById("email").value;
  let p = document.getElementById("pass").value;
  if(e == "" || p == "") {
    alert("Fields cannot be empty");
    return false;
  }
  // Simple mock credentials
  if(e == "user@test.com" && p == "12345") {
    document.cookie = "session=active; path=/";
    document.getElementById("status").innerHTML = "✅ Logged in!";
  } else {
    alert("Invalid login");
    return false;
  }
  return false; // prevent page refresh
}
// Clear cookie on logout
function logout() {
  document.cookie = "session=; max-age=0; path=/";
  document.getElementById("status").innerHTML = "🚪 Logged out!";
}
</script>
</body>
</html>
```

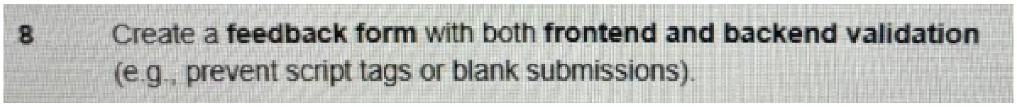


Login

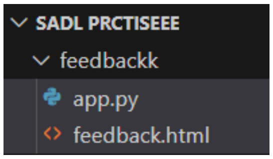
Email: user@test.com

Password: •••••

🚪 Logged out!



8 Create a **feedback form** with both **frontend and backend validation** (e.g., prevent script tags or blank submissions).



SADL PRCTISEEE

feedbackk

app.py

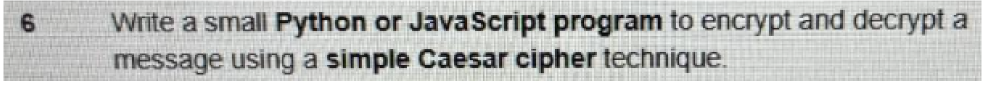
feedback.html

feedback.html

```
<!DOCTYPE html>
<html>
<body>
<h3>Feedback Form</h3>
<form method="POST" action="/submit" onsubmit="return check();">
  Name: <input id="name" name="name"><br><br>
  Feedback: <textarea id="fb" name="fb"></textarea><br><br>
  <button type="submit">Send</button>
</form>
<script>
function check(){
  if(name.value.trim()=="" || fb.value.trim()==""){
    alert("Fields cannot be empty");
    return false;
  }
  if(name.value.includes("<script") || fb.value.includes("<script")){
    alert("Script not allowed");
    return false;
  }
  return true; // allow submit
}
</script>
</body>
</html>
```

app.py

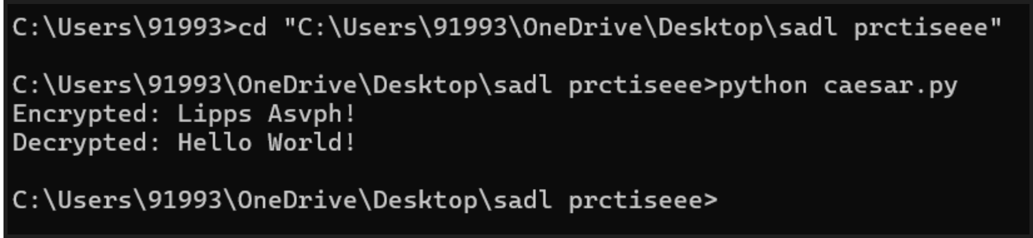
```
from flask import Flask, request
app = Flask(__name__)
@app.route("/")    # ✅ This will display the form
def home():
    return open("feedback.html").read()    # ✅ Loads your HTML file
@app.route("/submit", methods=["POST"])
def submit():
    name = request.form["name"].strip()
    fb = request.form["fb"].strip()
    if name == "" or fb == "":
        return "Error: Empty fields"
    if "<script" in name.lower() or "<script" in fb.lower():
        return "Error: Script tag not allowed"
    return "Feedback Received: " + name + " - " + fb
app.run()
```



6 Write a small **Python or JavaScript program** to encrypt and decrypt a message using a **simple Caesar cipher** technique.

caesar.py

```
def caesar_cipher(text, shift, mode='encrypt'):
    result = ''
    if mode == 'decrypt':
        shift = -shift
    for char in text:
        if char.isalpha():
            base = ord('A') if char.isupper() else ord('a')
            result += chr((ord(char) - base + shift) % 26 + base)
        else:
            result += char
    return result
# Example usage
message = "Hello World!"
shift = 4
encrypted = caesar_cipher(message, shift, 'encrypt')
decrypted = caesar_cipher(encrypted, shift, 'decrypt')
print("Encrypted:", encrypted)
print("Decrypted:", decrypted)
```



```
C:\Users\91993>cd "C:\Users\91993\OneDrive\Desktop\sadl prctiseee"

C:\Users\91993\OneDrive\Desktop\sadl prctiseee>python caesar.py
Encrypted: Lipps Asvph!
Decrypted: Hello World!

C:\Users\91993\OneDrive\Desktop\sadl prctiseee>
```

2 Demonstrate a **SQL Injection** vulnerability using a basic login form and then fix it by using **prepared statements**.

```
<!doctype html>
<html>
<head><meta charset="utf-8"><title>SQLi Demo (No DB)</title></head>
<body>
  <h3>Stored: admin / admin123</h3>
  <h4>Vulnerable (simulated)</h4>
  User: <input id="v_u"><br>
  Pass: <input id="v_p"><br>
  <button onclick="vuln()">Run</button>
  <pre id="v_out"></pre>
  <h4>Secure (simulated)</h4>
  User: <input id="s_u"><br>
  Pass: <input id="s_p"><br>
  <button onclick="sec()">Run</button>
  <pre id="s_out"></pre>
<script>
const storedUser = "admin";
const storedPass = "admin123";
function vuln(){
  const u = document.getElementById('v_u').value;
  const p = document.getElementById('v_p').value;
  const fake = `SELECT * FROM users WHERE username = '${u}' AND password = '${p}'`;
  let out = "Fake SQL: " + fake + "\n";
  if (fake.includes("' OR '1'='1")) out += "✅ Login SUCCESS (VULNERABLE - injection worked!)";
  else if (u===storedUser && p===storedPass) out += "✅ Login SUCCESS (correct creds)";
  else out += "❌ Login FAILED";
  document.getElementById('v_out').textContent = out;
}
function sec(){
  const u = document.getElementById('s_u').value;
  const p = document.getElementById('s_p').value;
  let out = "Secure check: direct value comparison\n";
  if (u===storedUser && p===storedPass) out += "✅ Login SUCCESS (SECURE)";
  else out += "❌ Login FAILED (SECURE)";
  document.getElementById('s_out').textContent = out;
}
</script>
</body>
</html>
```