

cookieforgeryattack.py (Page 1 of 1)

```
1: #!/usr/bin/env python
2:
3: #-----
4: # cookieforgeryattack.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import socket
10:
11: def main():
12:
13:     if len(sys.argv) != 3:
14:         print('Usage: python %s host port' % sys.argv[0])
15:         sys.exit(1)
16:
17:     try:
18:         host = sys.argv[1]
19:         port = int(sys.argv[2])
20:
21:         with socket.socket() as sock:
22:             sock.connect((host, port))
23:
24:             out_flo = sock.makefile(mode='w', encoding='iso-8859-1')
25:             out_flo.write('GET ' + '/show' + ' HTTP/1.1\r\n')
26:             out_flo.write('Host: ' + host + '\r\n')
27:             out_flo.write('Cookie: username=rdondero\r\n')
28:             out_flo.write('\r\n')
29:             out_flo.flush()
30:
31:             in_flo = sock.makefile(mode='r', encoding='iso-8859-1')
32:             for line in in_flo:
33:                 print(line, end='')
34:
35:     except Exception as ex:
36:         print(ex, file=sys.stderr)
37:         sys.exit(1)
38:
39: if __name__ == '__main__':
40:     main()
```

blank (Page 1 of 1)

```
1: This page is intentionally blank.
```

PennyAdmin07Encrypt/auth.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # auth.py
5: # Author: Bob Dondero
6: #-----
7:
8: import os
9: import cryptocode
10: import flask
11: import dotenv
12: import database
13:
14: from top import app
15:
16: #-----
17:
18: dotenv.load_dotenv()
19: secret_key = os.environ['APP_SECRET_KEY']
20:
21: #-----
22:
23: def _valid_username_and_password(username, password):
24:
25:     stored_password = database.get_password(username)
26:     if stored_password is None:
27:         return False
28:     return password == stored_password
29:
30: #-----
31:
32: @app.route('/login', methods=['GET'])
33: def login():
34:
35:     msg = flask.request.args.get('msg')
36:     if msg is None:
37:         msg = ''
38:     html = flask.render_template('login.html', msg=msg)
39:     response = flask.make_response(html)
40:     return response
41:
42: #-----
43:
44: @app.route('/handlelogin', methods=['POST'])
45: def handle_login():
46:
47:     username = flask.request.form.get('username')
48:     password = flask.request.form.get('password')
49:     if (username is None) or (username.strip() == ''):
50:         return flask.redirect(
51:             flask.url_for('login', msg='Wrong username or password'))
52:     if (password is None) or (password.strip() == ''):
53:         return flask.redirect(
54:             flask.url_for('login', msg='Wrong username or password'))
55:     if not _valid_username_and_password(username, password):
56:         return flask.redirect(
57:             flask.url_for('login', msg='Wrong username or password'))
58:     original_url = flask.request.cookies.get('original_url', '/index')
59:     response = flask.redirect(original_url)
60:     encrypted_username = cryptocode.encrypt(username, secret_key)
61:     response.set_cookie('username', encrypted_username)
62:     return response
63:
64: #-----
65:

```

PennyAdmin07Encrypt/auth.py (Page 2 of 2)

```

66: @app.route('/logout', methods=['GET'])
67: def logout():
68:
69:     html_code = flask.render_template('loggedout.html')
70:     response = flask.make_response(html_code)
71:
72:     # Delete cookies in the browser by setting them to expire at
73:     # a time that is in the past.
74:     response.set_cookie('username', '', expires=0)
75:     response.set_cookie('original_url', '', expires=0)
76:
77:     return response
78:
79: #-----
80:
81: @app.route('/signup', methods=['GET'])
82: def signup():
83:
84:     error_msg = flask.request.args.get('error_msg')
85:     if error_msg is None:
86:         error_msg = ''
87:     html_code = flask.render_template('signup.html',
88:         error_msg=error_msg)
89:     response = flask.make_response(html_code)
90:     return response
91:
92: #-----
93:
94: @app.route('/handlesignup', methods=['POST'])
95: def handle_signup():
96:
97:     username = flask.request.form.get('username')
98:     password = flask.request.form.get('password')
99:     if (username is None) or (username.strip() == ''):
100:         return flask.redirect(
101:             flask.url_for('signup', error_msg='Invalid username'))
102:     if (password is None) or (password.strip() == ''):
103:         return flask.redirect(
104:             flask.url_for('signup', error_msg='Invalid password'))
105:     successful = database.add_user(username, password)
106:     if not successful:
107:         return flask.redirect(
108:             flask.url_for('signup', error_msg='Duplicate username'))
109:     return flask.redirect(
110:         flask.url_for('login', msg='You now are signed up.))')
111:
112: #-----
113:
114: def authenticate():
115:
116:     encrypted_username = flask.request.cookies.get('username')
117:
118:     if encrypted_username is None:
119:         response = flask.redirect(flask.url_for('login'))
120:         response.set_cookie('original_url', flask.request.url)
121:         flask.abort(response)
122:
123:     username = cryptocode.decrypt(encrypted_username, secret_key)
124:     if not username:
125:         response = flask.redirect(flask.url_for('login'))
126:         response.set_cookie('original_url', flask.request.url)
127:         flask.abort(response)
128:
129:     return username

```

PennyAdmin08Session/top.py (Page 1 of 1)

```
1: #!/usr/bin/env python
2:
3: #-----
4: # top.py
5: # Author: Bob Dondero
6: #-----
7:
8: import os
9: import dotenv
10: import flask
11:
12: app = flask.Flask('penny', template_folder='.')
13:
14: dotenv.load_dotenv()
15: app.secret_key = os.environ['APP_SECRET_KEY']
```

blank (Page 1 of 1)

1: This page is intentionally blank.

PennyAdmin08Session/auth.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # auth.py
5: # Author: Bob Dondero
6: #-----
7:
8: import os
9: import flask
10: import dotenv
11: import database
12:
13: from top import app
14:
15: #-----
16:
17: def _valid_username_and_password(username, password):
18:
19:     stored_password = database.get_password(username)
20:     if stored_password is None:
21:         return False
22:     return password == stored_password
23:
24: #-----
25:
26: @app.route('/login', methods=['GET'])
27: def login():
28:
29:     msg = flask.request.args.get('msg')
30:     if msg is None:
31:         msg = ''
32:
33:     html = flask.render_template('login.html', msg=msg)
34:     response = flask.make_response(html)
35:     return response
36:
37: #-----
38:
39: @app.route('/handlelogin', methods=['POST'])
40: def handle_login():
41:
42:     username = flask.request.form.get('username')
43:     password = flask.request.form.get('password')
44:     if (username is None) or (username.strip() == ''):
45:         return flask.redirect(
46:             flask.url_for('login', msg='Wrong username or password'))
47:     if (password is None) or (password.strip() == ''):
48:         return flask.redirect(
49:             flask.url_for('login', msg='Wrong username or password'))
50:     if not _valid_username_and_password(username, password):
51:         return flask.redirect(
52:             flask.url_for('login', msg='Wrong username or password'))
53:     original_url = flask.session.get('original_url', '/index')
54:     response = flask.redirect(original_url)
55:     flask.session['username'] = username
56:     return response
57:
58: #-----
59:
60: @app.route('/logout', methods=['GET'])
61: def logout():
62:
63:     flask.session.clear()
64:     html_code = flask.render_template('loggedout.html')
65:     response = flask.make_response(html_code)

```

PennyAdmin08Session/auth.py (Page 2 of 2)

```

66:     return response
67:
68: #-----
69:
70: @app.route('/signup', methods=['GET'])
71: def signup():
72:
73:     error_msg = flask.request.args.get('error_msg')
74:     if error_msg is None:
75:         error_msg = ''
76:
77:     html_code = flask.render_template('signup.html',
78:         error_msg=error_msg)
79:     response = flask.make_response(html_code)
80:     return response
81:
82: #-----
83:
84: @app.route('/handlesignup', methods=['POST'])
85: def handle_signup():
86:
87:     username = flask.request.form.get('username')
88:     password = flask.request.form.get('password')
89:     if (username is None) or (username.strip() == ''):
90:         return flask.redirect(
91:             flask.url_for('signup', error_msg='Invalid username'))
92:     if (password is None) or (password.strip() == ''):
93:         return flask.redirect(
94:             flask.url_for('signup', error_msg='Invalid password'))
95:     successful = database.add_user(username, password)
96:     if not successful:
97:         return flask.redirect(
98:             flask.url_for('signup', error_msg='Duplicate username'))
99:
100:     return flask.redirect(
101:         flask.url_for('login', msg='You now are signed up.))
102:
103: #-----
104:
105: def authenticate():
106:
107:     username = flask.session.get('username')
108:     if username is None:
109:         response = flask.redirect(flask.url_for('login'))
110:         flask.session['original_url'] = flask.request.url
111:         flask.abort(response)
112:     return username

```

csrfattack.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>PennyAdmin</title>
5:   </head>
6:   <body>
7:     <hr>Hello, and welcome to <strong>PennyAdmin</strong><hr>
8:     <h1>Search for a Book by ISBN</h1>
9:     <form action="http://localhost:55555/handledelete" method="post">
10:      Enter an ISBN:
11:      <input type="text" name="isbn" autofocus>
12:      <input type="submit" value="Go">
13:    </form>
14:    <br>
15:    <br>
16:    <a href="http://localhost:55555/index">Return to home page</a>
17:    <br>
18:    <hr>
19:    <a href="http://localhost:55555/logout">Log out</a>
20:    of the application</br>
21:    <hr>
22:    Created by <a href="https://www.cs.princeton.edu/~rdondero">
23:      Bob Dondero</a>
24:    <hr>
25:  </body>
26: </html>

```

PennyAdmin09CsrfToken/add.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>PennyAdmin</title>
5:   </head>
6:   <body>
7:     {% include 'header.html' %}
8:     <h1>Add a Book</h1>
9:     <form action="/handleadd" method="post">
10:      <input type="hidden" name="csrf_token" value="{{csrf_token}}">
11:
12:      Enter an ISBN:
13:      <input type="text" name="isbn" autofocus
14:        required pattern=".*\S.*"
15:        title="At least one non-white-space char">
16:      <br>
17:
18:      Enter an author:
19:      <input type="text" name="author"
20:        required pattern=".*\S.*"
21:        title="At least one non-white-space char">
22:      <br>
23:
24:      Enter a title:
25:      <input type="text" name="title"
26:        required pattern=".*\S.*"
27:        title="At least one non-white-space char">
28:      <br>
29:
30:      <input type="submit" value="Go">
31:    </form>
32:    <br>
33:    <a href="/index">Return to home page</a>
34:    <br>
35:    {% include 'footer.html' %}
36:  </body>
37: </html>

```

PennyAdmin09CsrfToken/delete.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>PennyAdmin</title>
5:   </head>
6:   <body>
7:     {% include 'header.html' %}
8:     <h1>Delete a Book</h1>
9:     <form action="/handledelete" method="post">
10:      <input type="hidden" name="csrf_token" value="{{csrf_token}}">
11:
12:      Enter an ISBN:
13:      <input type="text" name="isbn" autofocus
14:        required pattern=".*\S.*"
15:        title="At least one non-white-space char">
16:      <input type="submit" value="Go">
17:    </form>
18:    <br>
19:    <br>
20:    <a href="/index">Return to home page</a>
21:    <br>
22:    {% include 'footer.html' %}
23:  </body>
24: </html>

```

PennyAdmin09CsrfToken/login.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:   </head>
6:   <body>
7:     <h1>Login to Penny</h1>
8:     <!-- Use post instead of get for security. -->
9:     <form action="/handlelogin" method="post">
10:      <input type="hidden" name="csrf_token" value="{{csrf_token}}">
11:
12:      <table>
13:        <tbody>
14:          <tr>
15:            <td style="text-align:right">User name:</td>
16:            <td><input type="text" name="username" autofocus
17:              required pattern=".*\S.*"
18:              title="At least one non-white-space char">
19:            </td>
20:          </tr>
21:          <tr>
22:            <td style="text-align:right">Password:</td>
23:            <td><input type="password" name="password"
24:              required pattern=".*\S.*"
25:              title="At least one non-white-space char">
26:            </td>
27:          </tr>
28:          <tr>
29:            <td></td>
30:            <td><input type="submit" value="Go"></td>
31:          </tr>
32:        </tbody>
33:      </table>
34:    </form>
35:    <br>
36:    Don't have an account? <a href="/signup">Sign up</a>
37:    <br>
38:    <br>
39:    <strong>{{msg}}</strong>
40:  </body>
41: </html>

```

PennyAdmin09CsrfToken/signup.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:   </head>
6:   <body>
7:     <h1>Penny New User Signup</h1>
8:     <!-- Use post instead of get for security. -->
9:     <form action="/handlesignup" method="post">
10:       <input type="hidden" name="csrf_token" value="{{csrf_token}}">
11:
12:       <table>
13:         <tbody>
14:           <tr>
15:             <td style="text-align:right">User name:</td>
16:             <td><input type="text" name="username" autofocus
17:               required pattern=".*\S.*"
18:               title="At least one non-white-space char">
19:             </td>
20:           </tr>
21:           <tr>
22:             <td style="text-align:right">Password:</td>
23:             <td><input type="password" name="password"
24:               required pattern=".*\S.*"
25:               title="At least one non-white-space char">
26:             <td>
27:           </tr>
28:           <tr>
29:             <td></td>
30:             <td><input type="submit" value="Go"></td>
31:           </tr>
32:         </tbody>
33:       </table>
34:     </form>
35:     <br>
36:     <strong>{{error_msg}}</strong>
37:   </body>
38: </html>

```

PennyAdmin09CsrfToken/penny.py (Page 1 of 3)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import os
9: import flask
10: import database
11: import auth
12:
13: from top import app
14:
15: #-----
16:
17: @app.route('/', methods=['GET'])
18: @app.route('/index', methods=['GET'])
19: def index():
20:
21:     username = auth.authenticate()
22:     is_authorized = database.is_authorized(username)
23:
24:     html_code = flask.render_template('index.html', username=username,
25:                                     is_authorized=is_authorized)
26:     response = flask.make_response(html_code)
27:     return response
28:
29: #-----
30:
31: @app.route('/show', methods=['GET'])
32: def show():
33:
34:     username = auth.authenticate()
35:
36:     books = database.get_books()
37:     html_code = flask.render_template('show.html',
38:                                     username=username, books=books)
39:     response = flask.make_response(html_code)
40:     return response
41:
42: #-----
43:
44: def report_results(username, message1, message2):
45:
46:     html_code = flask.render_template('reportresults.html',
47:                                     username=username, message1=message1, message2=message2)
48:     response = flask.make_response(html_code)
49:     return response
50:
51: #-----
52:
53: @app.route('/add', methods=['GET'])
54: def add():
55:
56:     username = auth.authenticate()
57:     if not database.is_authorized(username):
58:         html_code = 'You are not authorized to add books.'
59:         response = flask.make_response(html_code)
60:         return response
61:
62:     csrf_token = os.urandom(12).hex()
63:     flask.session['csrf_token'] = csrf_token
64:
65:     html_code = flask.render_template('add.html',

```

PennyAdmin09CsrfToken/penny.py (Page 2 of 3)

```

66:         username=username, csrf_token=csrf_token)
67:
68:     response = flask.make_response(html_code)
69:     return response
70:
71: #-----
72:
73: @app.route('/handleadd', methods=['POST'])
74: def handle_add():
75:
76:     username = auth.authenticate()
77:     if not database.is_authorized(username):
78:         html_code = 'You are not authorized to add books.'
79:         response = flask.make_response(html_code)
80:         return response
81:
82:     csrf_token_from_session = flask.session.get('csrf_token')
83:     csrf_token_from_request = flask.request.form.get('csrf_token')
84:
85:     if ((csrf_token_from_session is None)
86:         or (csrf_token_from_request is None)
87:         or (csrf_token_from_request != csrf_token_from_session)):
88:         html_code = '<title>400 Bad Request</title>'
89:         html_code += '<h1>Bad Request</h1>'
90:         html_code += '<p>The CSRF token is missing or bad.</p>'
91:         response = flask.make_response(html_code)
92:         return response
93:
94:     isbn = flask.request.form.get('isbn')
95:     if (isbn is None) or (isbn.strip() == ''):
96:         return report_results(username, 'Missing ISBN', '')
97:
98:     author = flask.request.form.get('author')
99:     if (author is None) or (author.strip() == ''):
100:         return report_results(username, 'Missing author', '')
101:
102:     title = flask.request.form.get('title')
103:     if (title is None) or (title.strip() == ''):
104:         return report_results(username, 'Missing title', '')
105:
106:     isbn = isbn.strip()
107:     author = author.strip()
108:     title = title.strip()
109:
110:     successful = database.add_book(isbn, author, title)
111:     if successful:
112:         message1 = 'The addition was successful'
113:         message2 = 'The database now contains a book with isbn ' + isbn
114:         message2 += ' author ' + author + ' and title ' + title
115:     else:
116:         message1 = 'The addition was unsuccessful'
117:         message2 = 'A book with ISBN ' + isbn + ' already exists'
118:
119:     return report_results(username, message1, message2)
120:
121: #-----
122:
123: @app.route('/delete', methods=['GET'])
124: def delete():
125:
126:     username = auth.authenticate()
127:     if not database.is_authorized(username):
128:         html_code = 'You are not authorized to delete books.'
129:         response = flask.make_response(html_code)
130:         return response

```

PennyAdmin09CsrfToken/penny.py (Page 3 of 3)

```

131:
132:     csrf_token = os.urandom(12).hex()
133:     flask.session['csrf_token'] = csrf_token
134:
135:     html_code = flask.render_template('delete.html',
136:                                       username=username, csrf_token=csrf_token)
137:
138:     response = flask.make_response(html_code)
139:     return response
140:
141: #-----
142:
143: @app.route('/handledelete', methods=['POST'])
144: def handle_delete():
145:
146:     username = auth.authenticate()
147:     if not database.is_authorized(username):
148:         html_code = 'You are not authorized to delete books.'
149:         response = flask.make_response(html_code)
150:         return response
151:
152:     csrf_token_from_session = flask.session.get('csrf_token')
153:     csrf_token_from_request = flask.request.form.get('csrf_token')
154:
155:     if ((csrf_token_from_session is None)
156:         or (csrf_token_from_request is None)
157:         or (csrf_token_from_request != csrf_token_from_session)):
158:         html_code = '<title>400 Bad Request</title>'
159:         html_code += '<h1>Bad Request</h1>'
160:         html_code += '<p>The CSRF token is missing or bad.</p>'
161:         response = flask.make_response(html_code)
162:         return response
163:
164:     isbn = flask.request.form.get('isbn')
165:     if (isbn is None) or (isbn.strip() == ''):
166:         return report_results(username, 'Missing ISBN', '')
167:
168:     isbn = isbn.strip()
169:
170:     database.delete_book(isbn)
171:
172:     message1 = 'The deletion was successful'
173:     message2 = 'The database now does not contain a book with ISBN '
174:     message2 += isbn
175:
176:     return report_results(username, message1, message2)

```

PennyAdmin09CsrfToken/auth.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # auth.py
5: # Author: Bob Dondero
6: #-----
7:
8: import os
9: import flask
10: import database
11: from top import app
12: #-----
13:
14: def _valid_username_and_password(username, password):
15:
16:     stored_password = database.get_password(username)
17:     if stored_password is None:
18:         return False
19:     return password == stored_password
20:
21: #-----
22:
23: @app.route('/login', methods=['GET'])
24: def login():
25:
26:     msg = flask.request.args.get('msg')
27:     if msg is None:
28:         msg = ''
29:
30:     csrf_token = os.urandom(12).hex()
31:     flask.session['csrf_token'] = csrf_token
32:
33:     html = flask.render_template('login.html',
34:                                 msg=msg, csrf_token=csrf_token)
35:
36:     response = flask.make_response(html)
37:     return response
38:
39: #-----
40:
41: @app.route('/handlelogin', methods=['POST'])
42: def handle_login():
43:
44:     csrf_token_from_session = flask.session.get('csrf_token')
45:     csrf_token_from_request = flask.request.form.get('csrf_token')
46:
47:     if ((csrf_token_from_session is None)
48:         or (csrf_token_from_request is None)
49:         or (csrf_token_from_request != csrf_token_from_session)):
50:         html_code = '<title>400 Bad Request</title>'
51:         html_code += '<h1>Bad Request</h1>'
52:         html_code += '<p>The CSRF token is missing or bad.</p>'
53:         response = flask.make_response(html_code)
54:         return response
55:
56:     username = flask.request.form.get('username')
57:     password = flask.request.form.get('password')
58:     if (username is None) or (username.strip() == ''):
59:         return flask.redirect(
60:             flask.url_for('login', msg='Wrong username or password'))
61:     if (password is None) or (password.strip() == ''):
62:         return flask.redirect(
63:             flask.url_for('login', msg='Wrong username or password'))
64:     if not _valid_username_and_password(username, password):
65:         return flask.redirect(

```

PennyAdmin09CsrfToken/auth.py (Page 2 of 2)

```

66:             flask.url_for('login', msg='Wrong username or password'))
67:     original_url = flask.session.get('original_url', '/index')
68:     response = flask.redirect(original_url)
69:     flask.session['username'] = username
70:     return response
71:
72: #-----
73:
74: @app.route('/logout', methods=['GET'])
75: def logout():
76:
77:     flask.session.clear()
78:     html_code = flask.render_template('loggedout.html')
79:     response = flask.make_response(html_code)
80:     return response
81:
82: #-----
83:
84: @app.route('/signup', methods=['GET'])
85: def signup():
86:
87:     error_msg = flask.request.args.get('error_msg')
88:     if error_msg is None:
89:         error_msg = ''
90:
91:     csrf_token = os.urandom(12).hex()
92:     flask.session['csrf_token'] = csrf_token
93:
94:     html_code = flask.render_template('signup.html',
95:                                     error_msg=error_msg, csrf_token=csrf_token)
96:
97:     response = flask.make_response(html_code)
98:     return response
99:
100: #-----
101:
102: @app.route('/handlesignup', methods=['POST'])
103: def handle_signup():
104:
105:     username = flask.request.form.get('username')
106:     password = flask.request.form.get('password')
107:     if (username is None) or (username.strip() == ''):
108:         return flask.redirect(
109:             flask.url_for('signup', error_msg='Invalid username'))
110:     if (password is None) or (password.strip() == ''):
111:         return flask.redirect(
112:             flask.url_for('signup', error_msg='Invalid password'))
113:     successful = database.add_user(username, password)
114:     if not successful:
115:         return flask.redirect(
116:             flask.url_for('signup', error_msg='Duplicate username'))
117:
118:     return flask.redirect(
119:         flask.url_for('login', msg='You now are signed up.))
120:
121: #-----
122:
123: def authenticate():
124:
125:     username = flask.session.get('username')
126:     if username is None:
127:         response = flask.redirect(flask.url_for('login'))
128:         flask.session['original_url'] = flask.request.url
129:         flask.abort(response)
130:     return username

```

PennyAdmin10Csrftoken/add.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>PennyAdmin</title>
5:   </head>
6:   <body>
7:     {% include 'header.html' %}
8:     <h1>Add a Book</h1>
9:     <form action="/handleadd" method="post">
10:      <input type="hidden" name="csrf_token"
11:        value="{{csrf_token()}}">
12:
13:      Enter an ISBN:
14:      <input type="text" name="isbn" autofocus
15:        required pattern=".*\S.*"
16:        title="At least one non-white-space char">
17:      <br>
18:
19:      Enter an author:
20:      <input type="text" name="author"
21:        required pattern=".*\S.*"
22:        title="At least one non-white-space char">
23:      <br>
24:
25:      Enter a title:
26:      <input type="text" name="title"
27:        required pattern=".*\S.*"
28:        title="At least one non-white-space char">
29:      <br>
30:
31:      <input type="submit" value="Go">
32:    </form>
33:    <br>
34:    <a href="/index">Return to home page</a>
35:    <br>
36:    {% include 'footer.html' %}
37:  </body>
38: </html>

```

PennyAdmin10Csrftoken/delete.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>PennyAdmin</title>
5:   </head>
6:   <body>
7:     {% include 'header.html' %}
8:     <h1>Delete a Book</h1>
9:     <form action="/handledelete" method="post">
10:      <input type="hidden" name="csrf_token"
11:        value="{{csrf_token()}}">
12:
13:      Enter an ISBN:
14:      <input type="text" name="isbn" autofocus
15:        required pattern=".*\S.*"
16:        title="At least one non-white-space char">
17:      <input type="submit" value="Go">
18:    </form>
19:    <br>
20:    <br>
21:    <a href="/index">Return to home page</a>
22:    <br>
23:    {% include 'footer.html' %}
24:  </body>
25: </html>

```

PennyAdmin10CsrfToken/login.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:   </head>
6:   <body>
7:     <h1>Login to Penny</h1>
8:     <!-- Use post instead of get for security. -->
9:     <form action="/handlelogin" method="post">
10:      <input type="hidden" name="csrf_token"
11:        value="{{csrf_token()}}">
12:
13:      <table>
14:        <tbody>
15:          <tr>
16:            <td style="text-align:right">User name:</td>
17:            <td><input type="text" name="username" autofocus
18:              required pattern=".*\S.*"
19:              title="At least one non-white-space char">
20:            </td>
21:          </tr>
22:          <tr>
23:            <td style="text-align:right">Password:</td>
24:            <td><input type="password" name="password"
25:              required pattern=".*\S.*"
26:              title="At least one non-white-space char">
27:            </td>
28:          </tr>
29:          <tr>
30:            <td></td>
31:            <td><input type="submit" value="Go"></td>
32:          </tr>
33:        </tbody>
34:      </table>
35:    </form>
36:    <br>
37:    Don't have an account? <a href="/signup">Sign up</a>
38:    <br>
39:    <br>
40:    <strong>{{msg}}</strong>
41:  </body>
42: </html>

```

PennyAdmin10CsrfToken/signup.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:   </head>
6:   <body>
7:     <h1>Penny New User Signup</h1>
8:     <!-- Use post instead of get for security. -->
9:     <form action="/handlesignup" method="post">
10:      <input type="hidden" name="csrf_token"
11:        value="{{csrf_token()}}">
12:
13:      <table>
14:        <tbody>
15:          <tr>
16:            <td style="text-align:right">User name:</td>
17:            <td><input type="text" name="username" autofocus
18:              required pattern=".*\S.*"
19:              title="At least one non-white-space char">
20:            </td>
21:          </tr>
22:          <tr>
23:            <td style="text-align:right">Password:</td>
24:            <td><input type="password" name="password"
25:              required pattern=".*\S.*"
26:              title="At least one non-white-space char">
27:            </td>
28:          </tr>
29:          <tr>
30:            <td></td>
31:            <td><input type="submit" value="Go"></td>
32:          </tr>
33:        </tbody>
34:      </table>
35:    </form>
36:    <br>
37:    <strong>{{error_msg}}</strong>
38:  </body>
39: </html>

```

PennyAdmin10CsrfToken/top.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # top.py
5: # Author: Bob Dondero
6: #-----
7:
8: import os
9: import flask
10: import flask_wtf.csrf
11: import dotenv
12:
13: app = flask.Flask('penny', template_folder='.')
14:
15: dotenv.load_dotenv()
16: app.secret_key = os.environ['APP_SECRET_KEY']
17:
18: flask_wtf.csrf.CSRFProtect(app)

```

PennyAdmin10CsrfToken/penny.py (Page 1 of 3)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import flask
9: import database
10: import auth
11:
12: from top import app
13:
14: #-----
15:
16: @app.route('/', methods=['GET'])
17: @app.route('/index', methods=['GET'])
18: def index():
19:
20:     username = auth.authenticate()
21:     is_authorized = auth.is_authorized(username)
22:
23:     html_code = flask.render_template('index.html', username=username,
24:                                     is_authorized=is_authorized)
25:     response = flask.make_response(html_code)
26:     return response
27:
28: #-----
29:
30: @app.route('/show', methods=['GET'])
31: def show():
32:
33:     username = auth.authenticate()
34:
35:     books = database.get_books()
36:     html_code = flask.render_template('show.html',
37:                                     username=username, books=books)
38:     response = flask.make_response(html_code)
39:     return response
40:
41: #-----
42:
43: def report_results(username, message1, message2):
44:
45:     html_code = flask.render_template('reportresults.html',
46:                                     username=username, message1=message1, message2=message2)
47:     response = flask.make_response(html_code)
48:     return response
49:
50: #-----
51:
52: @app.route('/add', methods=['GET'])
53: def add():
54:
55:     username = auth.authenticate()
56:     if not auth.is_authorized(username):
57:         html_code = 'You are not authorized to add books.'
58:         response = flask.make_response(html_code)
59:         return response
60:
61:     html_code = flask.render_template('add.html', username=username)
62:
63:     response = flask.make_response(html_code)
64:     return response
65:

```

PennyAdmin10CsrfToken/penny.py (Page 2 of 3)

```

66: #-----
67:
68: @app.route('/handleadd', methods=['POST'])
69: def handle_add():
70:
71:     username = auth.authenticate()
72:     if not auth.is_authorized(username):
73:         html_code = 'You are not authorized to add books.'
74:         response = flask.make_response(html_code)
75:         return response
76:
77:     isbn = flask.request.form.get('isbn')
78:     if (isbn is None) or (isbn.strip() == ''):
79:         return report_results(username, 'Missing ISBN', '')
80:
81:     author = flask.request.form.get('author')
82:     if (author is None) or (author.strip() == ''):
83:         return report_results(username, 'Missing author', '')
84:
85:     title = flask.request.form.get('title')
86:     if (title is None) or (title.strip() == ''):
87:         return report_results(username, 'Missing title', '')
88:
89:     isbn = isbn.strip()
90:     author = author.strip()
91:     title = title.strip()
92:
93:     successful = database.add_book(isbn, author, title)
94:     if successful:
95:         message1 = 'The addition was successful'
96:         message2 = 'The database now contains a book with isbn ' + isbn
97:         message2 += ' author ' + author + ' and title ' + title
98:     else:
99:         message1 = 'The addition was unsuccessful'
100:        message2 = 'A book with ISBN ' + isbn + ' already exists'
101:
102:        return report_results(username, message1, message2)
103:
104: #-----
105:
106: @app.route('/delete', methods=['GET'])
107: def delete():
108:
109:     username = auth.authenticate()
110:     if not auth.is_authorized(username):
111:         html_code = 'You are not authorized to delete books.'
112:         response = flask.make_response(html_code)
113:         return response
114:
115:     html_code = flask.render_template('delete.html', username=username)
116:
117:     response = flask.make_response(html_code)
118:     return response
119:
120: #-----
121:
122: @app.route('/handledelete', methods=['POST'])
123: def handle_delete():
124:
125:     username = auth.authenticate()
126:     if not auth.is_authorized(username):
127:         html_code = 'You are not authorized to delete books.'
128:         response = flask.make_response(html_code)
129:         return response
130:

```

PennyAdmin10CsrfToken/penny.py (Page 3 of 3)

```

131:     isbn = flask.request.form.get('isbn')
132:     if (isbn is None) or (isbn.strip() == ''):
133:         return report_results(username, 'Missing ISBN', '')
134:
135:     isbn = isbn.strip()
136:
137:     database.delete_book(isbn)
138:
139:     message1 = 'The deletion was successful'
140:     message2 = 'The database now does not contain a book with ISBN '
141:     message2 += isbn
142:
143:     return report_results(username, message1, message2)

```

PennyAdmin10Csrftoken/auth.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # auth.py
5: # Author: Bob Dondero
6: #-----
7:
8: import flask
9: import database
10:
11: from top import app
12:
13: #-----
14:
15: def _valid_username_and_password(username, password):
16:
17:     stored_password = database.get_password(username)
18:     if stored_password is None:
19:         return False
20:     return password == stored_password
21:
22: #-----
23:
24: @app.route('/login', methods=['GET'])
25: def login():
26:
27:     msg = flask.request.args.get('msg')
28:     if msg is None:
29:         msg = ''
30:
31:     html = flask.render_template('login.html', msg=msg)
32:
33:     response = flask.make_response(html)
34:     return response
35:
36: #-----
37:
38: @app.route('/handlelogin', methods=['POST'])
39: def handle_login():
40:
41:     username = flask.request.form.get('username')
42:     password = flask.request.form.get('password')
43:     if (username is None) or (username.strip() == ''):
44:         return flask.redirect(
45:             flask.url_for('login', msg='Wrong username or password'))
46:     if (password is None) or (password.strip() == ''):
47:         return flask.redirect(
48:             flask.url_for('login', msg='Wrong username or password'))
49:     if not _valid_username_and_password(username, password):
50:         return flask.redirect(
51:             flask.url_for('login', msg='Wrong username or password'))
52:     original_url = flask.session.get('original_url', '/index')
53:     response = flask.redirect(original_url)
54:     flask.session['username'] = username
55:     return response
56:
57: #-----
58:
59: @app.route('/logout', methods=['GET'])
60: def logout():
61:
62:     flask.session.clear()
63:     html_code = flask.render_template('loggedout.html')
64:     response = flask.make_response(html_code)
65:     return response

```

PennyAdmin10Csrftoken/auth.py (Page 2 of 2)

```

66:
67: #-----
68:
69: @app.route('/signup', methods=['GET'])
70: def signup():
71:
72:     error_msg = flask.request.args.get('error_msg')
73:     if error_msg is None:
74:         error_msg = ''
75:
76:     html_code = flask.render_template('signup.html',
77:         error_msg=error_msg)
78:
79:     response = flask.make_response(html_code)
80:     return response
81:
82: #-----
83:
84: @app.route('/handlesignup', methods=['POST'])
85: def handle_signup():
86:
87:     username = flask.request.form.get('username')
88:     password = flask.request.form.get('password')
89:     if (username is None) or (username.strip() == ''):
90:         return flask.redirect(
91:             flask.url_for('signup', error_msg='Invalid username'))
92:     if (password is None) or (password.strip() == ''):
93:         return flask.redirect(
94:             flask.url_for('signup', error_msg='Invalid password'))
95:     successful = database.add_user(username, password)
96:     if not successful:
97:         return flask.redirect(
98:             flask.url_for('signup', error_msg='Duplicate username'))
99:
100:     return flask.redirect(
101:         flask.url_for('login', msg='You now are signed up.))
102:     # return flask.redirect(flask.url_for('login'))
103:
104: #-----
105:
106: def authenticate():
107:
108:     username = flask.session.get('username')
109:     if username is None:
110:         response = flask.redirect(flask.url_for('login'))
111:         flask.session['original_url'] = flask.request.url
112:         flask.abort(response)
113:     return username
114:
115: #-----
116:
117: def is_authorized(username):
118:
119:     return database.is_authorized(username)

```

PennyAdmin11Hash/penny.sql (Page 1 of 1)

```
1: DROP TABLE IF EXISTS books;
2: CREATE TABLE books (isbn TEXT PRIMARY KEY, author TEXT, title TEXT);
3: INSERT INTO books (isbn, author, title)
4:   VALUES ('123', 'Kernighan', 'The Practice of Programming');
5: INSERT INTO books (isbn, author, title)
6:   VALUES ('234', 'Kernighan', 'The C Programming Language');
7: INSERT INTO books (isbn, author, title)
8:   VALUES ('345', 'Sedgewick', 'Algorithms in C');
9:
10: DROP TABLE IF EXISTS users;
11: CREATE TABLE users (username TEXT PRIMARY KEY, password TEXT);
12: INSERT INTO users (username, password) VALUES ('rdontero',
13:   'cd2eb0837c9b4c962c22d2ff8b5441b7b45805887f051d39bf133b583baf6860');
14: INSERT INTO users (username, password) VALUES ('bwk',
15:   'f2afd1cacb5441a5e65a7a460a5f9898b7b98b08aa6323a2e53c8b9a9686cd86');
16: INSERT INTO users (username, password) VALUES ('rs',
17:   '17f165d5a5ba695f27c023a83aa2b3463e23810e360b7517127e90161eebabda');
18:
19: DROP TABLE IF EXISTS authorizedusers;
20: CREATE TABLE authorizedusers (username TEXT PRIMARY KEY);
21: INSERT INTO authorizedusers (username) VALUES ('rdontero');
22: INSERT INTO authorizedusers (username) VALUES ('bwk');
```

blank (Page 1 of 1)

1: This page is intentionally blank.

PennyAdmin11Hash/auth.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # auth.py
5: # Author: Bob Dondero
6: #-----
7:
8: import hashlib
9: import flask
10: import database
11:
12: from top import app
13:
14: #-----
15:
16: def _hash(password):
17:
18:     hash_code = hashlib.sha256()
19:     hash_code.update(password.encode('ascii'))
20:     hash_code = hash_code.hexdigest()
21:     return hash_code
22:
23: #-----
24:
25: def _valid_username_and_password(username, password):
26:
27:     stored_hash_code = database.get_password(username)
28:     if stored_hash_code is None:
29:         return False
30:     hash_code = _hash(password)
31:     return hash_code == stored_hash_code
32:
33: #-----
34:
35: @app.route('/login', methods=['GET'])
36: def login():
37:
38:     msg = flask.request.args.get('msg')
39:     if msg is None:
40:         msg = ''
41:
42:     html = flask.render_template('login.html', msg=msg)
43:
44:     response = flask.make_response(html)
45:     return response
46:
47: #-----
48:
49: @app.route('/handlelogin', methods=['POST'])
50: def handle_login():
51:
52:     username = flask.request.form.get('username')
53:     password = flask.request.form.get('password')
54:     if (username is None) or (username.strip() == ''):
55:         return flask.redirect(
56:             flask.url_for('login', msg='Wrong username or password'))
57:     if (password is None) or (password.strip() == ''):
58:         return flask.redirect(
59:             flask.url_for('login', msg='Wrong username or password'))
60:     if not _valid_username_and_password(username, password):
61:         return flask.redirect(
62:             flask.url_for('login', msg='Wrong username or password'))
63:     original_url = flask.session.get('original_url', '/index')
64:     response = flask.redirect(original_url)
65:     flask.session['username'] = username

```

PennyAdmin11Hash/auth.py (Page 2 of 2)

```

66:     return response
67:
68: #-----
69:
70: @app.route('/logout', methods=['GET'])
71: def logout():
72:
73:     flask.session.clear()
74:     html_code = flask.render_template('loggedout.html')
75:     response = flask.make_response(html_code)
76:     return response
77:
78: #-----
79:
80: @app.route('/signup', methods=['GET'])
81: def signup():
82:
83:     error_msg = flask.request.args.get('error_msg')
84:     if error_msg is None:
85:         error_msg = ''
86:
87:     html_code = flask.render_template('signup.html',
88:                                     error_msg=error_msg)
89:
90:     response = flask.make_response(html_code)
91:     return response
92:
93: #-----
94:
95: @app.route('/handlesignup', methods=['POST'])
96: def handle_signup():
97:
98:     username = flask.request.form.get('username')
99:     password = flask.request.form.get('password')
100:    if (username is None) or (username.strip() == ''):
101:        return flask.redirect(
102:            flask.url_for('signup', error_msg='Invalid username'))
103:    if (password is None) or (password.strip() == ''):
104:        return flask.redirect(
105:            flask.url_for('signup', error_msg='Invalid password'))
106:
107:    hash_code = _hash(password)
108:    successful = database.add_user(username, hash_code)
109:    if not successful:
110:        return flask.redirect(
111:            flask.url_for('signup', error_msg='Duplicate username'))
112:
113:    return flask.redirect(
114:        flask.url_for('login', msg='You now are signed up.))
115:
116: #-----
117:
118: def authenticate():
119:
120:     username = flask.session.get('username')
121:     if username is None:
122:         response = flask.redirect(flask.url_for('login'))
123:         flask.session['original_url'] = flask.request.url
124:         flask.abort(response)
125:     return username

```

PennyAdmin12SaltHash/penny.sql (Page 1 of 1)

```
1: DROP TABLE IF EXISTS books;
2: CREATE TABLE books (isbn TEXT PRIMARY KEY, author TEXT, title TEXT);
3: INSERT INTO books (isbn, author, title)
4:   VALUES ('123', 'Kernighan', 'The Practice of Programming');
5: INSERT INTO books (isbn, author, title)
6:   VALUES ('234', 'Kernighan', 'The C Programming Language');
7: INSERT INTO books (isbn, author, title)
8:   VALUES ('345', 'Sedgewick', 'Algorithms in C');
9:
10: DROP TABLE IF EXISTS users;
11: CREATE TABLE users (username TEXT PRIMARY KEY, password TEXT);
12: INSERT INTO users (username, password) VALUES ('rdonero',
13:   'pbkdf2:sha256:600000$UOVCFEB4bjAW4nYx$0641776b12a4054fe5fb72eb4f9bbf
73c4266099de5ec01f09c325ed56746c3a');
14: INSERT INTO users (username, password) VALUES ('bwk',
15:   'pbkdf2:sha256:600000$fQwzUw8ZCPET43r$2390d394f64f239e5bc765f4163d49
d40b97601fae4d6ae0830c52296438e6ae');
16: INSERT INTO users (username, password) VALUES ('rs',
17:   'pbkdf2:sha256:600000$xbvn5wijiR8eJ5Mq$5bb39b39b45d65628091f4ba5fabda
99b30e9a60c447818db626f35f1dffdf9f');
18:
19: DROP TABLE IF EXISTS authorizedusers;
20: CREATE TABLE authorizedusers (username TEXT PRIMARY KEY);
21: INSERT INTO authorizedusers (username) VALUES ('rdonero');
22: INSERT INTO authorizedusers (username) VALUES ('bwk');
```

blank (Page 1 of 1)

1: This page is intentionally blank.

PennyAdmin12SaltHash/auth.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # auth.py
5: # Author: Bob Dondero
6: #-----
7:
8: import werkzeug.security
9: import flask
10: import database
11:
12: from top import app
13:
14: #-----
15:
16: def _valid_username_and_password(username, password):
17:
18:     stored_hash_code = database.get_password(username)
19:     if stored_hash_code is None:
20:         return False
21:     return werkzeug.security.check_password_hash(
22:         stored_hash_code, password)
23:
24: #-----
25:
26: @app.route('/login', methods=['GET'])
27: def login():
28:
29:     msg = flask.request.args.get('msg')
30:     if msg is None:
31:         msg = ''
32:
33:     html = flask.render_template('login.html', msg=msg)
34:
35:     response = flask.make_response(html)
36:     return response
37:
38: #-----
39:
40: @app.route('/handlelogin', methods=['POST'])
41: def handle_login():
42:
43:     username = flask.request.form.get('username')
44:     password = flask.request.form.get('password')
45:     if (username is None) or (username.strip() == ''):
46:         return flask.redirect(
47:             flask.url_for('login', msg='Wrong username or password'))
48:     if (password is None) or (password.strip() == ''):
49:         return flask.redirect(
50:             flask.url_for('login', msg='Wrong username or password'))
51:     if not _valid_username_and_password(username, password):
52:         return flask.redirect(
53:             flask.url_for('login', msg='Wrong username or password'))
54:     original_url = flask.session.get('original_url', '/index')
55:     response = flask.redirect(original_url)
56:     flask.session['username'] = username
57:     return response
58:
59: #-----
60:
61: @app.route('/logout', methods=['GET'])
62: def logout():
63:
64:     flask.session.clear()
65:     html_code = flask.render_template('loggedout.html')

```

PennyAdmin12SaltHash/auth.py (Page 2 of 2)

```

66:     response = flask.make_response(html_code)
67:     return response
68:
69: #-----
70:
71: @app.route('/signup', methods=['GET'])
72: def signup():
73:
74:     error_msg = flask.request.args.get('error_msg')
75:     if error_msg is None:
76:         error_msg = ''
77:
78:     html_code = flask.render_template('signup.html',
79:         error_msg=error_msg)
80:
81:     response = flask.make_response(html_code)
82:     return response
83:
84: #-----
85:
86: @app.route('/handlesignup', methods=['POST'])
87: def handle_signup():
88:
89:     username = flask.request.form.get('username')
90:     password = flask.request.form.get('password')
91:     if (username is None) or (username.strip() == ''):
92:         return flask.redirect(
93:             flask.url_for('signup', error_msg='Invalid username'))
94:     if (password is None) or (password.strip() == ''):
95:         return flask.redirect(
96:             flask.url_for('signup', error_msg='Invalid password'))
97:
98:     hash_code = werkzeug.security.generate_password_hash(
99:         password, 'pbkdf2')
100:     successful = database.add_user(username, hash_code)
101:     if not successful:
102:         return flask.redirect(
103:             flask.url_for('signup', error_msg='Duplicate username'))
104:
105:     return flask.redirect(
106:         flask.url_for('login', msg='You now are signed up.))
107:
108: #-----
109:
110: def authenticate():
111:
112:     username = flask.session.get('username')
113:     if username is None:
114:         response = flask.redirect(flask.url_for('login'))
115:         flask.session['original_url'] = flask.request.url
116:         flask.abort(response)
117:     return username

```