

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:             with sock.makefile(
25:                 mode='w', encoding='iso-8859-1') as out_flo:
26:                 out_flo.write('GET ' + '/show' + ' HTTP/1.1\r\n')
27:                 out_flo.write('Host: ' + host + '\r\n')
28:                 out_flo.write('Cookie: username=rdondero\r\n')
29:                 out_flo.write('\r\n')
30:
31:             with sock.makefile(
32:                 mode='r', encoding='iso-8859-1') as in_flo:
33:                 for line in in_flo:
34:                     print(line, end='')
35:
36:     except Exception as ex:
37:         print(ex, file=sys.stderr)
38:         sys.exit(1)
39:
40: if __name__ == '__main__':
41:     main()
```

blank (Page 1 of 1)

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

PennyAdmin05aEncrypt/auth.py (Page 1 of 3)

```

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: #-----
15:
16: dotenv.load_dotenv()
17: _APP_SECRET_KEY = os.environ['APP_SECRET_KEY']
18:
19: #-----
20: # Authentication routes
21: #-----
22:
23: def init(app):
24:
25:     app.add_url_rule('/login', 'login', login,
26:                     methods=['GET'])
27:     app.add_url_rule('/handlelogin', 'handle_login', handle_login,
28:                     methods=['POST'])
29:     app.add_url_rule('/logout', 'logout', logout,
30:                     methods=['GET'])
31:     app.add_url_rule('/signup', 'signup', signup,
32:                     methods=['GET'])
33:     app.add_url_rule('/handlesignup', 'handle_signup', handle_signup,
34:                     methods=['POST'])
35:
36: #-----
37:
38: def login():
39:
40:     msg = flask.request.args.get('msg')
41:     if msg is None:
42:         msg = ''
43:     html = flask.render_template('login.html', msg=msg)
44:     response = flask.make_response(html)
45:     return response
46:
47: #-----
48:
49: def handle_login():
50:
51:     username = flask.request.form.get('username')
52:     password = flask.request.form.get('password')
53:     if (username is None) or (username.strip() == ''):
54:         return flask.redirect(
55:             flask.url_for('login', msg='Wrong username or password'))
56:     if (password is None) or (password.strip() == ''):
57:         return flask.redirect(
58:             flask.url_for('login', msg='Wrong username or password'))
59:     if not _valid_username_and_password(username, password):
60:         return flask.redirect(
61:             flask.url_for('login', msg='Wrong username or password'))
62:     original_url = flask.request.cookies.get('original_url', '/index')
63:     response = flask.redirect(original_url)
64:     encrypted_username = cryptocode.encrypt(username, _APP_SECRET_KEY)
65:     response.set_cookie('username', encrypted_username)

```

PennyAdmin05aEncrypt/auth.py (Page 2 of 3)

```

66:     return response
67:
68: #-----
69:
70: def logout():
71:
72:     html_code = flask.render_template('loggedout.html')
73:     response = flask.make_response(html_code)
74:
75:     # Delete cookies in the browser by setting them to expire at
76:     # a time that is in the past.
77:     response.set_cookie('username', '', expires=0)
78:     response.set_cookie('original_url', '', expires=0)
79:
80:     return response
81:
82: #-----
83:
84: def signup():
85:
86:     error_msg = flask.request.args.get('error_msg')
87:     if error_msg is None:
88:         error_msg = ''
89:     html_code = flask.render_template('signup.html',
90:                                     error_msg=error_msg)
91:     response = flask.make_response(html_code)
92:     return response
93:
94: #-----
95:
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:    successful = database.add_user(username, password)
107:    if not successful:
108:        return flask.redirect(
109:            flask.url_for('signup', error_msg='Duplicate username'))
110:    return flask.redirect(
111:        flask.url_for('login', msg='You now are signed up.))
112:
113: #-----
114: # Authentication functions
115: #-----
116:
117: def _valid_username_and_password(username, password):
118:
119:     stored_password = database.get_password(username)
120:     if stored_password is None:
121:         return False
122:     return password == stored_password
123:
124: #-----
125:
126: def get_username():
127:
128:     encrypted_username = flask.request.cookies.get('username')
129:     if encrypted_username is None:
130:         return None

```

PennyAdmin05aEncrypt/auth.py (Page 3 of 3)

```
131:     username = cryptocode.decrypt(encrypted_username, _APP_SECRET_KEY)
132:     if not username:
133:         return None
134:     return username
135:
136: #-----
137:
138: def authenticate():
139:
140:     encrypted_username = flask.request.cookies.get('username')
141:
142:     if encrypted_username is None:
143:         response = flask.redirect(flask.url_for('login'))
144:         response.set_cookie('original_url', flask.request.url)
145:         flask.abort(response)
146:
147:     username = cryptocode.decrypt(encrypted_username, _APP_SECRET_KEY)
148:     if not username:
149:         response = flask.redirect(flask.url_for('login'))
150:         response.set_cookie('original_url', flask.request.url)
151:         flask.abort(response)
```

blank (Page 1 of 1)

1: This page is intentionally blank.

PennyAdmin05bSession/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 dotenv
10: import database
11: import auth
12:
13: #-----
14:
15: dotenv.load_dotenv()
16: _APP_SECRET_KEY = os.environ['APP_SECRET_KEY']
17:
18: #-----
19:
20: app = flask.Flask(__name__, template_folder='.')
21: app.secret_key = _APP_SECRET_KEY
22: auth.init(app)
23:
24: #-----
25:
26: @app.route('/', methods=['GET'])
27: @app.route('/index', methods=['GET'])
28: def index():
29:
30:     html_code = flask.render_template('index.html')
31:     response = flask.make_response(html_code)
32:     return response
33:
34: #-----
35:
36: @app.route('/menu', methods=['GET'])
37: def menu():
38:
39:     auth.authenticate()
40:     username = auth.get_username()
41:
42:     is_authorized = database.is_authorized(username)
43:
44:     html_code = flask.render_template('menu.html', username=username,
45:                                     is_authorized=is_authorized)
46:     response = flask.make_response(html_code)
47:     return response
48:
49: #-----
50:
51: @app.route('/show', methods=['GET'])
52: def show():
53:
54:     auth.authenticate()
55:     username = auth.get_username()
56:
57:     books = database.get_books()
58:     html_code = flask.render_template('show.html',
59:                                     username=username, books=books)
60:     response = flask.make_response(html_code)
61:     return response
62:
63: #-----
64:
65: @app.route('/add', methods=['GET'])

```

PennyAdmin05bSession/penny.py (Page 2 of 3)

```

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

```

PennyAdmin05bSession/penny.py (Page 3 of 3)

```
131:         response = flask.make_response(html_code)
132:         return response
133:
134:     html_code = flask.render_template('delete.html', username=username)
135:     response = flask.make_response(html_code)
136:     return response
137:
138: #-----
139:
140: @app.route('/handledelete', methods=['POST'])
141: def handle_delete():
142:
143:     auth.authenticate()
144:     username = auth.get_username()
145:
146:     if not database.is_authorized(username):
147:         html_code = 'You are not authorized to delete books.'
148:         response = flask.make_response(html_code)
149:         return response
150:
151:     isbn = flask.request.form.get('isbn')
152:     if (isbn is None) or (isbn.strip() == ''):
153:         message = 'The deletion was unsuccessful: missing ISBN'
154:     else:
155:         isbn = isbn.strip()
156:         database.delete_book(isbn)
157:         message = 'The deletion was successful'
158:
159:     html_code = flask.render_template('reportresults.html',
160:                                     username=username, message=message)
161:     response = flask.make_response(html_code)
162:     return response
```

blank (Page 1 of 1)

1: This page is intentionally blank.

PennyAdmin05bSession/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:
12: #-----
13: # Authentication routes
14: #-----
15:
16: def init(app):
17:
18:     app.add_url_rule('/login', 'login', login,
19:         methods=['GET'])
20:     app.add_url_rule('/handlelogin', 'handle_login', handle_login,
21:         methods=['POST'])
22:     app.add_url_rule('/logout', 'logout', logout,
23:         methods=['GET'])
24:     app.add_url_rule('/signup', 'signup', signup,
25:         methods=['GET'])
26:     app.add_url_rule('/handlesignup', 'handle_signup', handle_signup,
27:         methods=['POST'])
28:
29: #-----
30:
31: def login():
32:
33:     msg = flask.request.args.get('msg')
34:     if msg is None:
35:         msg = ''
36:     html = flask.render_template('login.html', msg=msg)
37:     response = flask.make_response(html)
38:     return response
39:
40: #-----
41:
42: def handle_login():
43:
44:     username = flask.request.form.get('username')
45:     password = flask.request.form.get('password')
46:     if (username is None) or (username.strip() == ''):
47:         return flask.redirect(
48:             flask.url_for('login', msg='Wrong username or password'))
49:     if (password is None) or (password.strip() == ''):
50:         return flask.redirect(
51:             flask.url_for('login', msg='Wrong username or password'))
52:     if not _valid_username_and_password(username, password):
53:         return flask.redirect(
54:             flask.url_for('login', msg='Wrong username or password'))
55:     original_url = flask.session.get('original_url', '/index')
56:     response = flask.redirect(original_url)
57:     flask.session['username'] = username
58:     return response
59:
60: #-----
61:
62: def logout():
63:
64:     flask.session.clear()
65:     html_code = flask.render_template('loggedout.html')

```

PennyAdmin05bSession/auth.py (Page 2 of 2)

```

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

```

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/menu">Return to menu 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>
```

blank (Page 1 of 1)

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

PennyAdmin06aCsrfToken/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="/menu">Return to menu page</a>
34:     <br>
35:     {% include 'footer.html' %}
36:   </body>
37: </html>

```

PennyAdmin06aCsrfToken/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="/menu">Return to menu page</a>
21:     <br>
22:     {% include 'footer.html' %}
23:   </body>
24: </html>

```

PennyAdmin06aCsrfToken/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>

```

PennyAdmin06aCsrfToken/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>

```

PennyAdmin06aCsrfToken/penny.py (Page 1 of 4)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import os
9: import flask
10: import dotenv
11: import database
12: import auth
13:
14: #-----
15:
16: dotenv.load_dotenv()
17: _APP_SECRET_KEY = os.environ['_APP_SECRET_KEY']
18:
19: #-----
20:
21: app = flask.Flask(__name__, template_folder='.')
22: app.secret_key = _APP_SECRET_KEY
23: auth.init(app)
24:
25: #-----
26:
27: @app.route('/', methods=['GET'])
28: @app.route('/index', methods=['GET'])
29: def index():
30:
31:     html_code = flask.render_template('index.html')
32:     response = flask.make_response(html_code)
33:     return response
34:
35: #-----
36:
37: @app.route('/menu', methods=['GET'])
38: def menu():
39:
40:     auth.authenticate()
41:     username = auth.get_username()
42:
43:     is_authorized = database.is_authorized(username)
44:
45:     html_code = flask.render_template('menu.html', username=username,
46:                                     is_authorized=is_authorized)
47:     response = flask.make_response(html_code)
48:     return response
49:
50: #-----
51:
52: @app.route('/show', methods=['GET'])
53: def show():
54:
55:     auth.authenticate()
56:     username = auth.get_username()
57:
58:     books = database.get_books()
59:     html_code = flask.render_template('show.html',
60:                                     username=username, books=books)
61:     response = flask.make_response(html_code)
62:     return response
63:
64: #-----
65:

```

PennyAdmin06aCsrfToken/penny.py (Page 2 of 4)

```

66: @app.route('/add', methods=['GET'])
67: def add():
68:
69:     auth.authenticate()
70:     username = auth.get_username()
71:
72:     if not database.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:     csrf_token = os.urandom(12).hex()
78:     flask.session['csrf_token'] = csrf_token
79:
80:     html_code = flask.render_template('add.html',
81:                                     username=username, csrf_token=csrf_token)
82:
83:     response = flask.make_response(html_code)
84:     return response
85:
86: #-----
87:
88: @app.route('/handleadd', methods=['POST'])
89: def handle_add():
90:
91:     auth.authenticate()
92:     username = auth.get_username()
93:
94:     if not database.is_authorized(username):
95:         html_code = 'You are not authorized to add books.'
96:         response = flask.make_response(html_code)
97:         return response
98:
99:     csrf_token_from_session = flask.session.get('csrf_token')
100:    csrf_token_from_request = flask.request.form.get('csrf_token')
101:
102:    if ((csrf_token_from_session is None)
103:        or (csrf_token_from_request is None)
104:        or (csrf_token_from_request != csrf_token_from_session)):
105:        html_code = '''
106:            <title>400 Bad Request</title>
107:            <h1>Bad Request</h1>
108:            <p>The CSRF token is missing or bad.</p>
109:            '''
110:        response = flask.make_response(html_code)
111:        return response
112:
113:    isbn = flask.request.form.get('isbn')
114:    if (isbn is None) or (isbn.strip() == ''):
115:        message = 'The addition was unsuccessful: missing ISBN'
116:    else:
117:        author = flask.request.form.get('author')
118:        if (author is None) or (author.strip() == ''):
119:            message = 'The addition was unsuccessful: missing author'
120:    else:
121:        title = flask.request.form.get('title')
122:        if (title is None) or (title.strip() == ''):
123:            message = 'The addition was unsuccessful: missing title'
124:    else:
125:        isbn = isbn.strip()
126:        author = author.strip()
127:        title = title.strip()
128:
129:        successful = database.add_book(isbn, author, title)
130:        if not successful:

```

PennyAdmin06aCsrfToken/penny.py (Page 3 of 4)

```

131:         message = 'The addition was unsuccessful: '
132:         message += 'duplicate ISBN'
133:     else:
134:         message = 'The addition was successful'
135:
136:     html_code = flask.render_template('reportresults.html',
137:         username=username, message=message)
138:     response = flask.make_response(html_code)
139:     return response
140:
141: #-----
142:
143: @app.route('/delete', methods=['GET'])
144: def delete():
145:
146:     auth.authenticate()
147:     username = auth.get_username()
148:
149:     if not database.is_authorized(username):
150:         html_code = 'You are not authorized to delete books.'
151:         response = flask.make_response(html_code)
152:         return response
153:
154:     csrf_token = os.urandom(12).hex()
155:     flask.session['csrf_token'] = csrf_token
156:
157:     html_code = flask.render_template('delete.html',
158:         username=username, csrf_token=csrf_token)
159:
160:     response = flask.make_response(html_code)
161:     return response
162:
163: #-----
164:
165: @app.route('/handledelete', methods=['POST'])
166: def handle_delete():
167:
168:     auth.authenticate()
169:     username = auth.get_username()
170:
171:     if not database.is_authorized(username):
172:         html_code = 'You are not authorized to delete books.'
173:         response = flask.make_response(html_code)
174:         return response
175:
176:     csrf_token_from_session = flask.session.get('csrf_token')
177:     csrf_token_from_request = flask.request.form.get('csrf_token')
178:
179:     if ((csrf_token_from_session is None)
180:         or (csrf_token_from_request is None)
181:         or (csrf_token_from_request != csrf_token_from_session)):
182:         html_code = '''
183:             <title>400 Bad Request</title>
184:             <h1>Bad Request</h1>
185:             <p>The CSRF token is missing or bad.</p>
186:             '''
187:         response = flask.make_response(html_code)
188:         return response
189:
190:     isbn = flask.request.form.get('isbn')
191:     if (isbn is None) or (isbn.strip() == ''):
192:         message = 'The deletion was unsuccessful: missing ISBN'
193:     else:
194:         isbn = isbn.strip()
195:         database.delete_book(isbn)

```

PennyAdmin06aCsrfToken/penny.py (Page 4 of 4)

```

196:         message = 'The deletion was successful'
197:
198:     html_code = flask.render_template('reportresults.html',
199:         username=username, message=message)
200:     response = flask.make_response(html_code)
201:     return response

```

PennyAdmin06aCsrfToken/auth.py (Page 1 of 3)

```

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:
12: #-----
13: # Authentication routes
14: #-----
15:
16: def init(app):
17:
18:     app.add_url_rule('/login', 'login', login,
19:                     methods=['GET'])
20:     app.add_url_rule('/handlelogin', 'handle_login', handle_login,
21:                     methods=['POST'])
22:     app.add_url_rule('/logout', 'logout', logout,
23:                     methods=['GET'])
24:     app.add_url_rule('/signup', 'signup', signup,
25:                     methods=['GET'])
26:     app.add_url_rule('/handlesignup', 'handle_signup', handle_signup,
27:                     methods=['POST'])
28:
29: #-----
30:
31: def login():
32:
33:     msg = flask.request.args.get('msg')
34:     if msg is None:
35:         msg = ''
36:
37:     csrf_token = os.urandom(12).hex()
38:     flask.session['csrf_token'] = csrf_token
39:
40:     html = flask.render_template('login.html',
41:                                msg=msg, csrf_token=csrf_token)
42:
43:     response = flask.make_response(html)
44:     return response
45:
46: #-----
47:
48: def handle_login():
49:
50:     csrf_token_from_session = flask.session.get('csrf_token')
51:     csrf_token_from_request = flask.request.form.get('csrf_token')
52:
53:     if ((csrf_token_from_session is None)
54:         or (csrf_token_from_request is None)
55:         or (csrf_token_from_request != csrf_token_from_session)):
56:         html_code = '''
57:         <title>400 Bad Request</title>
58:         <h1>Bad Request</h1>
59:         <p>The CSRF token is missing or bad.</p>
60:         '''
61:         response = flask.make_response(html_code)
62:         return response
63:
64:     username = flask.request.form.get('username')
65:     password = flask.request.form.get('password')

```

PennyAdmin06aCsrfToken/auth.py (Page 2 of 3)

```

66:     if (username is None) or (username.strip() == ''):
67:         return flask.redirect(
68:             flask.url_for('login', msg='Wrong username or password'))
69:     if (password is None) or (password.strip() == ''):
70:         return flask.redirect(
71:             flask.url_for('login', msg='Wrong username or password'))
72:     if not _valid_username_and_password(username, password):
73:         return flask.redirect(
74:             flask.url_for('login', msg='Wrong username or password'))
75:     original_url = flask.session.get('original_url', '/index')
76:     response = flask.redirect(original_url)
77:     flask.session['username'] = username
78:     return response
79:
80: #-----
81:
82: def logout():
83:
84:     flask.session.clear()
85:     html_code = flask.render_template('loggedout.html')
86:     response = flask.make_response(html_code)
87:     return response
88:
89: #-----
90:
91: def signup():
92:
93:     error_msg = flask.request.args.get('error_msg')
94:     if error_msg is None:
95:         error_msg = ''
96:
97:     csrf_token = os.urandom(12).hex()
98:     flask.session['csrf_token'] = csrf_token
99:
100:    html_code = flask.render_template('signup.html',
101:                                     error_msg=error_msg, csrf_token=csrf_token)
102:
103:    response = flask.make_response(html_code)
104:    return response
105:
106: #-----
107:
108: def handle_signup():
109:
110:    csrf_token_from_session = flask.session.get('csrf_token')
111:    csrf_token_from_request = flask.request.form.get('csrf_token')
112:
113:    if ((csrf_token_from_session is None)
114:        or (csrf_token_from_request is None)
115:        or (csrf_token_from_request != csrf_token_from_session)):
116:        html_code = '''
117:        <title>400 Bad Request</title>
118:        <h1>Bad Request</h1>
119:        <p>The CSRF token is missing or bad.</p>
120:        '''
121:        response = flask.make_response(html_code)
122:        return response
123:
124:    username = flask.request.form.get('username')
125:    password = flask.request.form.get('password')
126:    if (username is None) or (username.strip() == ''):
127:        return flask.redirect(
128:            flask.url_for('signup', error_msg='Invalid username'))
129:    if (password is None) or (password.strip() == ''):
130:        return flask.redirect(

```

PennyAdmin06aCsrfToken/auth.py (Page 3 of 3)

```
131:         flask.url_for('signup', error_msg='Invalid password'))
132:     successful = database.add_user(username, password)
133:     if not successful:
134:         return flask.redirect(
135:             flask.url_for('signup', error_msg='Duplicate username'))
136:     return flask.redirect(
137:         flask.url_for('login', msg='You now are signed up.'))
138:
139: #-----
140: # Authentication functions
141: #-----
142:
143: def _valid_username_and_password(username, password):
144:
145:     stored_password = database.get_password(username)
146:     if stored_password is None:
147:         return False
148:     return password == stored_password
149:
150: #-----
151:
152: def get_username():
153:
154:     return flask.session.get('username')
155:
156: #-----
157:
158: def authenticate():
159:
160:     username = flask.session.get('username')
161:     if username is None:
162:         response = flask.redirect(flask.url_for('login'))
163:         flask.session['original_url'] = flask.request.url
164:         flask.abort(response)
```

blank (Page 1 of 1)

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

PennyAdmin06bCsrfToken/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="/menu">Return to menu page</a>
35:    <br>
36:    {% include 'footer.html' %}
37:  </body>
38: </html>

```

PennyAdmin06bCsrfToken/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="/menu">Return to menu page</a>
22:    <br>
23:    {% include 'footer.html' %}
24:  </body>
25: </html>

```

PennyAdmin06bCsrfToken/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>

```

PennyAdmin06bCsrfToken/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>

```

PennyAdmin06bCsrfToken/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 dotenv
10: import flask
11: import flask_wtf.csrf
12: import database
13: import auth
14:
15: #-----
16:
17: dotenv.load_dotenv()
18: _APP_SECRET_KEY = os.environ['APP_SECRET_KEY']
19:
20: #-----
21:
22: app = flask.Flask(__name__, template_folder='.')
23: app.secret_key = _APP_SECRET_KEY
24: auth.init(app)
25: flask_wtf.csrf.CSRFProtect(app)
26:
27: #-----
28:
29: @app.route('/', methods=['GET'])
30: @app.route('/index', methods=['GET'])
31: def index():
32:
33:     html_code = flask.render_template('index.html')
34:     response = flask.make_response(html_code)
35:     return response
36:
37: #-----
38:
39: @app.route('/menu', methods=['GET'])
40: def menu():
41:
42:     auth.authenticate()
43:     username = auth.get_username()
44:
45:     is_authorized = database.is_authorized(username)
46:
47:     html_code = flask.render_template('menu.html', username=username,
48:                                     is_authorized=is_authorized)
49:     response = flask.make_response(html_code)
50:     return response
51:
52: #-----
53:
54: @app.route('/show', methods=['GET'])
55: def show():
56:
57:     auth.authenticate()
58:     username = auth.get_username()
59:
60:     books = database.get_books()
61:     html_code = flask.render_template('show.html',
62:                                     username=username, books=books)
63:     response = flask.make_response(html_code)
64:     return response
65:

```

PennyAdmin06bCsrfToken/penny.py (Page 2 of 3)

```

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

```

PennyAdmin06bCsrfToken/penny.py (Page 3 of 3)

```
131:     username = auth.get_username()
132:
133:     if not database.is_authorized(username):
134:         html_code = 'You are not authorized to delete books.'
135:         response = flask.make_response(html_code)
136:         return response
137:
138:     html_code = flask.render_template('delete.html', username=username)
139:
140:     response = flask.make_response(html_code)
141:     return response
142:
143: #-----
144:
145: @app.route('/handledelete', methods=['POST'])
146: def handle_delete():
147:
148:     auth.authenticate()
149:     username = auth.get_username()
150:
151:     if not database.is_authorized(username):
152:         html_code = 'You are not authorized to delete books.'
153:         response = flask.make_response(html_code)
154:         return response
155:
156:     isbn = flask.request.form.get('isbn')
157:     if (isbn is None) or (isbn.strip() == ''):
158:         message = 'The deletion was unsuccessful: missing ISBN'
159:     else:
160:         isbn = isbn.strip()
161:         database.delete_book(isbn)
162:         message = 'The deletion was successful'
163:
164:     html_code = flask.render_template('reportresults.html',
165:         username=username, message=message)
166:     response = flask.make_response(html_code)
167:     return response
```

blank (Page 1 of 1)

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

PennyAdmin06bCsrfToken/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: #-----
12: # Authentication routes
13: #-----
14:
15: def init(app):
16:
17:     app.add_url_rule('/login', 'login', login,
18:                     methods=['GET'])
19:     app.add_url_rule('/handlelogin', 'handle_login', handle_login,
20:                     methods=['POST'])
21:     app.add_url_rule('/logout', 'logout', logout,
22:                     methods=['GET'])
23:     app.add_url_rule('/signup', 'signup', signup,
24:                     methods=['GET'])
25:     app.add_url_rule('/handlesignup', 'handle_signup', handle_signup,
26:                     methods=['POST'])
27:
28: #-----
29:
30: def login():
31:
32:     msg = flask.request.args.get('msg')
33:     if msg is None:
34:         msg = ''
35:
36:     html = flask.render_template('login.html', msg=msg)
37:
38:     response = flask.make_response(html)
39:     return response
40:
41: #-----
42:
43: def handle_login():
44:
45:     username = flask.request.form.get('username')
46:     password = flask.request.form.get('password')
47:     if (username is None) or (username.strip() == ''):
48:         return flask.redirect(
49:             flask.url_for('login', msg='Wrong username or password'))
50:     if (password is None) or (password.strip() == ''):
51:         return flask.redirect(
52:             flask.url_for('login', msg='Wrong username or password'))
53:     if not _valid_username_and_password(username, password):
54:         return flask.redirect(
55:             flask.url_for('login', msg='Wrong username or password'))
56:     original_url = flask.session.get('original_url', '/index')
57:     response = flask.redirect(original_url)
58:     flask.session['username'] = username
59:     return response
60:
61: #-----
62:
63: def logout():
64:
65:     flask.session.clear()

```

PennyAdmin06bCsrfToken/auth.py (Page 2 of 2)

```

66:     html_code = flask.render_template('loggedout.html')
67:     response = flask.make_response(html_code)
68:     return response
69:
70: #-----
71:
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: def handle_signup():
87:
88:     username = flask.request.form.get('username')
89:     password = flask.request.form.get('password')
90:     if (username is None) or (username.strip() == ''):
91:         return flask.redirect(
92:             flask.url_for('signup', error_msg='Invalid username'))
93:     if (password is None) or (password.strip() == ''):
94:         return flask.redirect(
95:             flask.url_for('signup', error_msg='Invalid password'))
96:     successful = database.add_user(username, password)
97:     if not successful:
98:         return flask.redirect(
99:             flask.url_for('signup', error_msg='Duplicate username'))
100:    return flask.redirect(
101:        flask.url_for('login', msg='You now are signed up.))
102:
103: #-----
104: # Authentication functions
105: #-----
106:
107: def _valid_username_and_password(username, password):
108:
109:     stored_password = database.get_password(username)
110:     if stored_password is None:
111:         return False
112:     return password == stored_password
113:
114: #-----
115:
116: def get_username():
117:
118:     return flask.session.get('username')
119:
120: #-----
121:
122: def authenticate():
123:
124:     username = flask.session.get('username')
125:     if username is None:
126:         response = flask.redirect(flask.url_for('login'))
127:         flask.session['original_url'] = flask.request.url
128:         flask.abort(response)

```