cookieforgeryattack.py (Page 1 of 1)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# cookieforgeryattack.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import sys
import socket

def main():

    if len(sys.argv) != 3:
        print('Usage: python %s host port' % sys.argv[0])
        sys.exit(1)

    try:
        host = sys.argv[1]
        port = int(sys.argv[2])

        with socket.socket() as sock:
            sock.connect((host, port))

            with sock.makefile(
                mode='w', encoding='iso-8859-1') as out_flo:
                out_flo.write('GET ' + '/show' + ' HTTP/1.1\r\n')
                out_flo.write('Host: ' + host + '\r\n')
                out_flo.write('Cookie: username=rdondero\r\n')
                out_flo.write('\r\n')

            with sock.makefile(
                mode='r', encoding='iso-8859-1') as in_flo:
                for line in in_flo:
                    print(line, end='')

    except Exception as ex:
        print(ex, file=sys.stderr)
        sys.exit(1)

if __name__ == '__main__':
    main()
```

blank (Page 1 of 1)

```
This page is intentionally blank.
```

PennyAdmin05aEncrypt/auth.py (Page 1 of 3)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# auth.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import os
import cryptocode
import flask
import dotenv
import database

#-----------------------------------------------------------------------

dotenv.load_dotenv()
_APP_SECRET_KEY = os.environ['APP_SECRET_KEY']

#-----------------------------------------------------------------------
# Authentication routes
#-----------------------------------------------------------------------

def init(app):

    app.add_url_rule('/login',  'login', login,
        methods=['GET'])
    app.add_url_rule('/handlelogin',  'handle_login', handle_login,
        methods=['POST'])
    app.add_url_rule('/logout',  'logout', logout,
        methods=['GET'])
    app.add_url_rule('/signup',  'signup', signup,
        methods=['GET'])
    app.add_url_rule('/handlesignup',  'handle_signup', handle_signup,
        methods=['POST'])

#-----------------------------------------------------------------------

def login():

    msg = flask.request.args.get('msg')
    if msg is None:
        msg = ''
    html = flask.render_template('login.html', msg=msg)
    response = flask.make_response(html)
    return response

#-----------------------------------------------------------------------

def handle_login():

    username = flask.request.form.get('username')
    password = flask.request.form.get('password')
    if (username is None) or (username.strip() == ''):
        return flask.redirect(
            flask.url_for('login', msg='Wrong username or password'))
    if (password is None) or (password.strip() == ''):
        return flask.redirect(
            flask.url_for('login', msg='Wrong username or password'))
    if not _valid_username_and_password(username, password):
        return flask.redirect(
            flask.url_for('login', msg='Wrong username or password'))
    original_url = flask.request.cookies.get('original_url', '/index')
    response = flask.redirect(original_url)
    encrypted_username = cryptocode.encrypt(username, _APP_SECRET_KEY)
    response.set_cookie('username', encrypted_username)
```

PennyAdmin05aEncrypt/auth.py (Page 2 of 3)

```
    return response

#-----------------------------------------------------------------------

def logout():

    html_code = flask.render_template('loggedout.html')
    response = flask.make_response(html_code)

    # Delete cookies in the browser by setting them to expire at
    # a time that is in the past.
    response.set_cookie('username', '', expires=0)
    response.set_cookie('original_url', '', expires=0)

    return response

#-----------------------------------------------------------------------

def signup():

    error_msg = flask.request.args.get('error_msg')
    if error_msg is None:
        error_msg = ''
    html_code = flask.render_template('signup.html',
        error_msg=error_msg)
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

def handle_signup():

    username = flask.request.form.get('username')
    password = flask.request.form.get('password')
    if (username is None) or (username.strip() == ''):
        return flask.redirect(
            flask.url_for('signup', error_msg='Invalid username'))
    if (password is None) or (password.strip() == ''):
        return flask.redirect(
            flask.url_for('signup', error_msg='Invalid password'))
    successful = database.add_user(username, password)
    if not successful:
        return flask.redirect(
            flask.url_for('signup', error_msg='Duplicate username'))
    return flask.redirect(
        flask.url_for('login', msg='You now are signed up.'))

#-----------------------------------------------------------------------
# Authentication functions
#-----------------------------------------------------------------------

def _valid_username_and_password(username, password):

    stored_password = database.get_password(username)
    if stored_password is None:
        return False
    return password == stored_password

#-----------------------------------------------------------------------

def get_username():

    encrypted_username = flask.request.cookies.get('username')
    if encrypted_username is None:
        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)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# penny.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import flask
import dotenv
import database
import auth

#-----------------------------------------------------------------------

dotenv.load_dotenv()
_APP_SECRET_KEY = os.environ['APP_SECRET_KEY']

#-----------------------------------------------------------------------

app = flask.Flask(__name__, template_folder='.')
app.secret_key = _APP_SECRET_KEY
auth.init(app)

#-----------------------------------------------------------------------

@app.route('/', methods=['GET'])
@app.route('/index', methods=['GET'])
def index():

    html_code = flask.render_template('index.html')
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/menu', methods=['GET'])
def menu():

    auth.authenticate()
    username = auth.get_username()

    is_authorized = database.is_authorized(username)

    html_code = flask.render_template('menu.html', username=username,
        is_authorized=is_authorized)
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/show', methods=['GET'])
def show():

    auth.authenticate()
    username = auth.get_username()

    books = database.get_books()
    html_code = flask.render_template('show.html',
        username=username, books=books)
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/add', methods=['GET'])
```

PennyAdmin05bSession/penny.py (Page 2 of 3)

```
def add():

    auth.authenticate()
    username = auth.get_username()

    if not database.is_authorized(username):
        html_code = 'You are not authorized to add books.'
        response = flask.make_response(html_code)
        return response

    html_code = flask.render_template('add.html', username=username)
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/handleadd', methods=['POST'])
def handle_add():

    auth.authenticate()
    username = auth.get_username()

    if not database.is_authorized(username):
        html_code = 'You are not authorized to add books.'
        response = flask.make_response(html_code)
        return response

    isbn = flask.request.form.get('isbn')
    if (isbn is None) or (isbn.strip() == ''):
        message = 'The addition was unsuccessful: missing ISBN'
    else:
        author = flask.request.form.get('author')
        if (author is None) or (author.strip() == ''):
            message = 'The addition was unsuccessful: missing author'
        else:
            title = flask.request.form.get('title')
            if (title is None) or (title.strip() == ''):
                message = 'The addition was unsuccessful: missing title'
            else:
                isbn = isbn.strip()
                author = author.strip()
                title = title.strip()

                successful = database.add_book(isbn, author, title)
                if not successful:
                    message = 'The addition was unsuccessful: '
                    message += 'duplicate ISBN'
                else:
                    message = 'The addition was successful'

    html_code = flask.render_template('reportresults.html',
        username=username, message=message)
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/delete', methods=['GET'])
def delete():

    auth.authenticate()
    username = auth.get_username()

    if not database.is_authorized(username):
        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)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# auth.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import os
import flask
import database

#-----------------------------------------------------------------------
# Authentication routes
#-----------------------------------------------------------------------

def init(app):

    app.add_url_rule('/login',  'login', login,
        methods=['GET'])
    app.add_url_rule('/handlelogin',  'handle_login', handle_login,
        methods=['POST'])
    app.add_url_rule('/logout',  'logout', logout,
        methods=['GET'])
    app.add_url_rule('/signup',  'signup', signup,
        methods=['GET'])
    app.add_url_rule('/handlesignup',  'handle_signup', handle_signup,
        methods=['POST'])

#-----------------------------------------------------------------------

def login():

    msg = flask.request.args.get('msg')
    if msg is None:
        msg = ''
    html = flask.render_template('login.html', msg=msg)
    response = flask.make_response(html)
    return response

#-----------------------------------------------------------------------

def handle_login():

    username = flask.request.form.get('username')
    password = flask.request.form.get('password')
    if (username is None) or (username.strip() == ''):
        return flask.redirect(
            flask.url_for('login', msg='Wrong username or password'))
    if (password is None) or (password.strip() == ''):
        return flask.redirect(
            flask.url_for('login', msg='Wrong username or password'))
    if not _valid_username_and_password(username, password):
        return flask.redirect(
            flask.url_for('login', msg='Wrong username or password'))
    original_url = flask.session.get('original_url', '/index')
    response = flask.redirect(original_url)
    flask.session['username'] = username
    return response

#-----------------------------------------------------------------------

def logout():

    flask.session.clear()
    html_code = flask.render_template('loggedout.html')
```

PennyAdmin05bSession/auth.py (Page 2 of 2)

```
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

def signup():

    error_msg = flask.request.args.get('error_msg')
    if error_msg is None:
        error_msg = ''
    html_code = flask.render_template('signup.html',
        error_msg=error_msg)
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

def handle_signup():

    username = flask.request.form.get('username')
    password = flask.request.form.get('password')
    if (username is None) or (username.strip() == ''):
        return flask.redirect(
            flask.url_for('signup', error_msg='Invalid username'))
    if (password is None) or (password.strip() == ''):
        return flask.redirect(
            flask.url_for('signup', error_msg='Invalid password'))
    successful = database.add_user(username, password)
    if not successful:
        return flask.redirect(
            flask.url_for('signup', error_msg='Duplicate username'))
    return flask.redirect(
        flask.url_for('login', msg='You now are signed up.'))

#-----------------------------------------------------------------------
# Authentication functions
#-----------------------------------------------------------------------

def _valid_username_and_password(username, password):

    stored_password = database.get_password(username)
    if stored_password is None:
        return False
    return password == stored_password

#-----------------------------------------------------------------------

def get_username():

    return flask.session.get('username')

#-----------------------------------------------------------------------

def authenticate():

    username = flask.session.get('username')
    if username is None:
        response = flask.redirect(flask.url_for('login'))
        flask.session['original_url'] = flask.request.url
        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)

```
<!DOCTYPE html>
<html>
   <head>
      <title>PennyAdmin</title>
   </head>
   <body>
      {% include 'header.html' %}
      <h1>Add a Book</h1>
      <form action="/handleadd" method="post">
         <input type="hidden" name="csrf_token" value="{{csrf_token}}">

         Enter an ISBN:
         <input type="text" name="isbn" autofocus
            required pattern=".*\S.*"
            title="At least one non-white-space char">
         <br>

         Enter an author:
         <input type="text" name="author"
            required pattern=".*\S.*"
            title="At least one non-white-space char">
         <br>

         Enter a title:
         <input type="text" name="title"
            required pattern=".*\S.*"
            title="At least one non-white-space char">
         <br>

         <input type="submit" value="Go">
      </form>
      <br>
      <a href="/menu">Return to menu page</a>
      <br>
      {% include 'footer.html' %}
   </body>
</html>
```

PennyAdmin06aCsrfToken/delete.html (Page 1 of 1)

```
<!DOCTYPE html>
<html>
   <head>
      <title>PennyAdmin</title>
   </head>
   <body>
      {% include 'header.html' %}
      <h1>Delete a Book</h1>
      <form action="/handledelete" method="post">
         <input type="hidden" name="csrf_token" value="{{csrf_token}}">

         Enter an ISBN:
         <input type="text" name="isbn" autofocus
            required pattern=".*\S.*"
            title="At least one non-white-space char">
         <input type="submit" value="Go">
      </form>
      <br>
      <br>
      <a href="/menu">Return to menu page</a>
      <br>
      {% include 'footer.html' %}
   </body>
</html>
```

PennyAdmin06aCsrfToken/login.html (Page 1 of 1)

```
<!DOCTYPE html>
<html>
   <head>
      <title>Penny.com</title>
   </head>
   <body>
      <h1>Login to Penny</h1>
      <!-- Use post instead of get for security. -->
      <form action="/handlelogin" method="post">
         <input type="hidden" name="csrf_token" value="{{csrf_token}}">

         <table>
            <tbody>
               <tr>
                  <td style="text-align:right">User name:</td>
                  <td><input type="text" name="username" autofocus
                     required pattern=".*\S.*"
                     title="At least one non-white-space char">
                  </td>
               </tr>
               <tr>
                  <td style="text-align:right">Password:</td>
                  <td><input type="password" name="password"
                     required pattern=".*\S.*"
                     title="At least one non-white-space char">
                  <td>
               </tr>
               <tr>
                  <td></td>
                  <td><input type="submit" value="Go"></td>
               </tr>
            </tbody>
         </table>
      </form>
      <br>
      Don't have an account? <a href="/signup">Sign up</a>
      <br>
      <br>
      <strong>{{msg}}</strong>
   </body>
</html>
```

PennyAdmin06aCsrfToken/signup.html (Page 1 of 1)

```
<!DOCTYPE html>
<html>
   <head>
      <title>Penny.com</title>
   </head>
   <body>
      <h1>Penny New User Signup</h1>
      <!-- Use post instead of get for security. -->
      <form action="/handlesignup" method="post">
         <input type="hidden" name="csrf_token" value="{{csrf_token}}">

         <table>
            <tbody>
               <tr>
                  <td style="text-align:right">User name:</td>
                  <td><input type="text" name="username" autofocus
                     required pattern=".*\S.*"
                     title="At least one non-white-space char">
                  </td>
               </tr>
               <tr>
                  <td style="text-align:right">Password:</td>
                  <td><input type="password" name="password"
                     required pattern=".*\S.*"
                     title="At least one non-white-space char">
                  <td>
               </tr>
               <tr>
                  <td></td>
                  <td><input type="submit" value="Go"></td>
               </tr>
            </tbody>
         </table>
      </form>
      <br>
      <strong>{{error_msg}}</strong>
   </body>
</html>
```

PennyAdmin06aCsrfToken/penny.py (Page 1 of 4)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# penny.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import os
import flask
import dotenv
import database
import auth

#-----------------------------------------------------------------------

dotenv.load_dotenv()
_APP_SECRET_KEY = os.environ['APP_SECRET_KEY']

#-----------------------------------------------------------------------

app = flask.Flask(__name__, template_folder='.')
app.secret_key = _APP_SECRET_KEY
auth.init(app)

#-----------------------------------------------------------------------

@app.route('/', methods=['GET'])
@app.route('/index', methods=['GET'])
def index():

    html_code = flask.render_template('index.html')
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/menu', methods=['GET'])
def menu():

    auth.authenticate()
    username = auth.get_username()

    is_authorized = database.is_authorized(username)

    html_code = flask.render_template('menu.html', username=username,
        is_authorized=is_authorized)
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/show', methods=['GET'])
def show():

    auth.authenticate()
    username = auth.get_username()

    books = database.get_books()
    html_code = flask.render_template('show.html',
        username=username, books=books)
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

```

PennyAdmin06aCsrfToken/penny.py (Page 2 of 4)

```
@app.route('/add', methods=['GET'])
def add():

    auth.authenticate()
    username = auth.get_username()

    if not database.is_authorized(username):
        html_code = 'You are not authorized to add books.'
        response = flask.make_response(html_code)
        return response

    csrf_token = os.urandom(12).hex()
    flask.session['csrf_token'] = csrf_token

    html_code = flask.render_template('add.html',
        username=username, csrf_token=csrf_token)

    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/handleadd', methods=['POST'])
def handle_add():

    auth.authenticate()
    username = auth.get_username()

    if not database.is_authorized(username):
        html_code = 'You are not authorized to add books.'
        response = flask.make_response(html_code)
        return response

    csrf_token_from_session = flask.session.get('csrf_token')
    csrf_token_from_request = flask.request.form.get('csrf_token')

    if ((csrf_token_from_session is None)
        or (csrf_token_from_request is None)
        or (csrf_token_from_request != csrf_token_from_session)):
        html_code = '''
            <title>400 Bad Request</title>
            <h1>Bad Request</h1>
            <p>The CSRF token is missing or bad.</p>
            '''
        response = flask.make_response(html_code)
        return response

    isbn = flask.request.form.get('isbn')
    if (isbn is None) or (isbn.strip() == ''):
        message = 'The addition was unsuccessful: missing ISBN'
    else:
        author = flask.request.form.get('author')
        if (author is None) or (author.strip() == ''):
            message = 'The addition was unsuccessful: missing author'
        else:
            title = flask.request.form.get('title')
            if (title is None) or (title.strip() == ''):
                message = 'The addition was unsuccessful: missing title'
            else:
                isbn = isbn.strip()
                author = author.strip()
                title = title.strip()

                successful = database.add_book(isbn, author, title)
                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)

```
#!/usr/bin/env python

#-------------------------------------------------------------------------
# auth.py
# Author: Bob Dondero
#-------------------------------------------------------------------------

import os
import flask
import database

#-------------------------------------------------------------------------
# Authentication routes
#-------------------------------------------------------------------------

def init(app):

    app.add_url_rule('/login',  'login', login,
        methods=['GET'])
    app.add_url_rule('/handlelogin',  'handle_login', handle_login,
        methods=['POST'])
    app.add_url_rule('/logout',  'logout', logout,
        methods=['GET'])
    app.add_url_rule('/signup',  'signup', signup,
        methods=['GET'])
    app.add_url_rule('/handlesignup',  'handle_signup', handle_signup,
        methods=['POST'])

#-------------------------------------------------------------------------

def login():

    msg = flask.request.args.get('msg')
    if msg is None:
        msg = ''

    csrf_token = os.urandom(12).hex()
    flask.session['csrf_token'] = csrf_token

    html = flask.render_template('login.html',
        msg=msg, csrf_token=csrf_token)

    response = flask.make_response(html)
    return response

#-------------------------------------------------------------------------

def handle_login():

    csrf_token_from_session = flask.session.get('csrf_token')
    csrf_token_from_request = flask.request.form.get('csrf_token')

    if ((csrf_token_from_session is None)
        or (csrf_token_from_request is None)
        or (csrf_token_from_request != csrf_token_from_session)):
        html_code = '''
            <title>400 Bad Request</title>
            <h1>Bad Request</h1>
            <p>The CSRF token is missing or bad.</p>
            '''
        response = flask.make_response(html_code)
        return response

    username = flask.request.form.get('username')
    password = flask.request.form.get('password')
```

PennyAdmin06aCsrfToken/auth.py (Page 2 of 3)

```
    if (username is None) or (username.strip() == ''):
        return flask.redirect(
            flask.url_for('login', msg='Wrong username or password'))
    if (password is None) or (password.strip() == ''):
        return flask.redirect(
            flask.url_for('login', msg='Wrong username or password'))
    if not _valid_username_and_password(username, password):
        return flask.redirect(
            flask.url_for('login', msg='Wrong username or password'))
    original_url = flask.session.get('original_url', '/index')
    response = flask.redirect(original_url)
    flask.session['username'] = username
    return response

#-------------------------------------------------------------------------

def logout():

    flask.session.clear()
    html_code = flask.render_template('loggedout.html')
    response = flask.make_response(html_code)
    return response

#-------------------------------------------------------------------------

def signup():

    error_msg = flask.request.args.get('error_msg')
    if error_msg is None:
        error_msg = ''

    csrf_token = os.urandom(12).hex()
    flask.session['csrf_token'] = csrf_token

    html_code = flask.render_template('signup.html',
        error_msg=error_msg, csrf_token=csrf_token)

    response = flask.make_response(html_code)
    return response

#-------------------------------------------------------------------------

def handle_signup():

    csrf_token_from_session = flask.session.get('csrf_token')
    csrf_token_from_request = flask.request.form.get('csrf_token')

    if ((csrf_token_from_session is None)
        or (csrf_token_from_request is None)
        or (csrf_token_from_request != csrf_token_from_session)):
        html_code = '''
            <title>400 Bad Request</title>
            <h1>Bad Request</h1>
            <p>The CSRF token is missing or bad.</p>
            '''
        response = flask.make_response(html_code)
        return response

    username = flask.request.form.get('username')
    password = flask.request.form.get('password')
    if (username is None) or (username.strip() == ''):
        return flask.redirect(
            flask.url_for('signup', error_msg='Invalid username'))
    if (password is None) or (password.strip() == ''):
        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)

```
<!DOCTYPE html>
<html>
   <head>
      <title>PennyAdmin</title>
   </head>
   <body>
      {% include 'header.html' %}
      <h1>Add a Book</h1>
      <form action="/handleadd" method="post">
         <input type="hidden" name="csrf_token"
            value="{{csrf_token()}}">

         Enter an ISBN:
         <input type="text" name="isbn" autofocus
            required pattern=".*\S.*"
            title="At least one non-white-space char">
         <br>

         Enter an author:
         <input type="text" name="author"
            required pattern=".*\S.*"
            title="At least one non-white-space char">
         <br>

         Enter a title:
         <input type="text" name="title"
            required pattern=".*\S.*"
            title="At least one non-white-space char">
         <br>

         <input type="submit" value="Go">
      </form>
      <br>
      <a href="/menu">Return to menu page</a>
      <br>
      {% include 'footer.html' %}
   </body>
</html>
```

PennyAdmin06bCsrfToken/delete.html (Page 1 of 1)

```
<!DOCTYPE html>
<html>
   <head>
      <title>PennyAdmin</title>
   </head>
   <body>
      {% include 'header.html' %}
      <h1>Delete a Book</h1>
      <form action="/handledelete" method="post">
         <input type="hidden" name="csrf_token"
            value="{{csrf_token()}}">

         Enter an ISBN:
         <input type="text" name="isbn" autofocus
            required pattern=".*\S.*"
            title="At least one non-white-space char">
         <input type="submit" value="Go">
      </form>
      <br>
      <a href="/menu">Return to menu page</a>
      <br>
      {% include 'footer.html' %}
   </body>
</html>
```

PennyAdmin06bCsrfToken/login.html (Page 1 of 1)

```
<!DOCTYPE html>
<html>
   <head>
      <title>Penny.com</title>
   </head>
   <body>
      <h1>Login to Penny</h1>
      <!-- Use post instead of get for security. -->
      <form action="/handlelogin" method="post">
         <input type="hidden" name="csrf_token"
            value="{{csrf_token()}}">

         <table>
            <tbody>
               <tr>
                  <td style="text-align:right">User name:</td>
                  <td><input type="text" name="username" autofocus
                     required pattern=".*\S.*"
                     title="At least one non-white-space char">
                  </td>
               </tr>
               <tr>
                  <td style="text-align:right">Password:</td>
                  <td><input type="password" name="password"
                     required pattern=".*\S.*"
                     title="At least one non-white-space char">
                  <td>
               </tr>
               <tr>
                  <td></td>
                  <td><input type="submit" value="Go"></td>
               </tr>
            </tbody>
         </table>
      </form>
      <br>
      Don't have an account? <a href="/signup">Sign up</a>
      <br>
      <br>
      <strong>{{msg}}</strong>
   </body>
</html>
```

PennyAdmin06bCsrfToken/signup.html (Page 1 of 1)

```
<!DOCTYPE html>
<html>
   <head>
      <title>Penny.com</title>
   </head>
   <body>
      <h1>Penny New User Signup</h1>
      <!-- Use post instead of get for security. -->
      <form action="/handlesignup" method="post">
         <input type="hidden" name="csrf_token"
            value="{{csrf_token()}}">

         <table>
            <tbody>
               <tr>
                  <td style="text-align:right">User name:</td>
                  <td><input type="text" name="username" autofocus
                     required pattern=".*\S.*"
                     title="At least one non-white-space char">
                  </td>
               </tr>
               <tr>
                  <td style="text-align:right">Password:</td>
                  <td><input type="password" name="password"
                     required pattern=".*\S.*"
                     title="At least one non-white-space char">
                  <td>
               </tr>
               <tr>
                  <td></td>
                  <td><input type="submit" value="Go"></td>
               </tr>
            </tbody>
         </table>
      </form>
      <br>
      <strong>{{error_msg}}</strong>
   </body>
</html>
```

PennyAdmin06bCsrfToken/penny.py (Page 1 of 3)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# penny.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import os
import dotenv
import flask
import flask_wtf.csrf
import database
import auth

#-----------------------------------------------------------------------

dotenv.load_dotenv()
_APP_SECRET_KEY = os.environ['APP_SECRET_KEY']

#-----------------------------------------------------------------------

app = flask.Flask(__name__, template_folder='.')
app.secret_key = _APP_SECRET_KEY
auth.init(app)
flask_wtf.csrf.CSRFProtect(app)

#-----------------------------------------------------------------------

@app.route('/', methods=['GET'])
@app.route('/index', methods=['GET'])
def index():

    html_code = flask.render_template('index.html')
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/menu', methods=['GET'])
def menu():

    auth.authenticate()
    username = auth.get_username()

    is_authorized = database.is_authorized(username)

    html_code = flask.render_template('menu.html', username=username,
        is_authorized=is_authorized)
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/show', methods=['GET'])
def show():

    auth.authenticate()
    username = auth.get_username()

    books = database.get_books()
    html_code = flask.render_template('show.html',
        username=username, books=books)
    response = flask.make_response(html_code)
    return response

```

PennyAdmin06bCsrfToken/penny.py (Page 2 of 3)

```
#-----------------------------------------------------------------------

@app.route('/add', methods=['GET'])
def add():

    auth.authenticate()
    username = auth.get_username()

    if not database.is_authorized(username):
        html_code = 'You are not authorized to add books.'
        response = flask.make_response(html_code)
        return response

    html_code = flask.render_template('add.html', username=username)

    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/handleadd', methods=['POST'])
def handle_add():

    auth.authenticate()
    username = auth.get_username()

    if not database.is_authorized(username):
        html_code = 'You are not authorized to add books.'
        response = flask.make_response(html_code)
        return response

    isbn = flask.request.form.get('isbn')
    if (isbn is None) or (isbn.strip() == ''):
        message = 'The addition was unsuccessful: missing ISBN'
    else:
        author = flask.request.form.get('author')
        if (author is None) or (author.strip() == ''):
            message = 'The addition was unsuccessful: missing author'
        else:
            title = flask.request.form.get('title')
            if (title is None) or (title.strip() == ''):
                message = 'The addition was unsuccessful: missing title'
            else:
                isbn = isbn.strip()
                author = author.strip()
                title = title.strip()

                successful = database.add_book(isbn, author, title)
                if not successful:
                    message = 'The addition was unsuccessful: '
                    message += 'duplicate ISBN'
                else:
                    message = 'The addition was successful'

    html_code = flask.render_template('reportresults.html',
        username=username, message=message)
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/delete', methods=['GET'])
def delete():

    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)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# auth.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import flask
import database

#-----------------------------------------------------------------------
# Authentication routes
#-----------------------------------------------------------------------

def init(app):

    app.add_url_rule('/login',  'login', login,
        methods=['GET'])
    app.add_url_rule('/handlelogin',  'handle_login', handle_login,
        methods=['POST'])
    app.add_url_rule('/logout',  'logout', logout,
        methods=['GET'])
    app.add_url_rule('/signup',  'signup', signup,
        methods=['GET'])
    app.add_url_rule('/handlesignup',  'handle_signup', handle_signup,
        methods=['POST'])

#-----------------------------------------------------------------------

def login():

    msg = flask.request.args.get('msg')
    if msg is None:
        msg = ''

    html = flask.render_template('login.html', msg=msg)

    response = flask.make_response(html)
    return response

#-----------------------------------------------------------------------

def handle_login():

    username = flask.request.form.get('username')
    password = flask.request.form.get('password')
    if (username is None) or (username.strip() == ''):
        return flask.redirect(
            flask.url_for('login', msg='Wrong username or password'))
    if (password is None) or (password.strip() == ''):
        return flask.redirect(
            flask.url_for('login', msg='Wrong username or password'))
    if not _valid_username_and_password(username, password):
        return flask.redirect(
            flask.url_for('login', msg='Wrong username or password'))
    original_url = flask.session.get('original_url', '/index')
    response = flask.redirect(original_url)
    flask.session['username'] = username
    return response

#-----------------------------------------------------------------------

def logout():

    flask.session.clear()
```

PennyAdmin06bCsrfToken/auth.py (Page 2 of 2)

```
    html_code = flask.render_template('loggedout.html')
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

def signup():

    error_msg = flask.request.args.get('error_msg')
    if error_msg is None:
        error_msg = ''

    html_code = flask.render_template('signup.html',
        error_msg=error_msg)

    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

def handle_signup():

    username = flask.request.form.get('username')
    password = flask.request.form.get('password')
    if (username is None) or (username.strip() == ''):
        return flask.redirect(
            flask.url_for('signup', error_msg='Invalid username'))
    if (password is None) or (password.strip() == ''):
        return flask.redirect(
            flask.url_for('signup', error_msg='Invalid password'))
    successful = database.add_user(username, password)
    if not successful:
        return flask.redirect(
            flask.url_for('signup', error_msg='Duplicate username'))
    return flask.redirect(
        flask.url_for('login', msg='You now are signed up.'))

#-----------------------------------------------------------------------
# Authentication functions
#-----------------------------------------------------------------------

def _valid_username_and_password(username, password):

    stored_password = database.get_password(username)
    if stored_password is None:
        return False
    return password == stored_password

#-----------------------------------------------------------------------

def get_username():

    return flask.session.get('username')

#-----------------------------------------------------------------------

def authenticate():

    username = flask.session.get('username')
    if username is None:
        response = flask.redirect(flask.url_for('login'))
        flask.session['original_url'] = flask.request.url
        flask.abort(response)
```