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

from top import app

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

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

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

def _valid_username_and_password(username, password):

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

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

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

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

@app.route('/handlelogin', methods=['POST'])
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, secret_key)
    response.set_cookie('username', encrypted_username)
    return response

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

```

PennyAdmin05aEncrypt/auth.py (Page 2 of 3)

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

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

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

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

@app.route('/handlesignup', methods=['POST'])
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.'))

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

def get_username():

    encrypted_username = flask.request.cookies.get('username')
    if encrypted_username is None:
        return None
    username = cryptocode.decrypt(encrypted_username, secret_key)
    if not username:
        return None
    return username

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

def authenticate():

    encrypted_username = flask.request.cookies.get('username')

    if encrypted_username is None:
```

PennyAdmin05aEncrypt/auth.py (Page 3 of 3)

```
131:         response = flask.redirect(flask.url_for('login'))
132:         response.set_cookie('original_url', flask.request.url)
133:         flask.abort(response)
134:
135:     username = cryptocode.decrypt(encrypted_username, secret_key)
136:     if not username:
137:         response = flask.redirect(flask.url_for('login'))
138:         response.set_cookie('original_url', flask.request.url)
139:         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 os
import dotenv
import flask
import database
import auth

from top import app

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

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

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

@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'])
def add():

    auth.authenticate()
```

PennyAdmin05bSession/penny.py (Page 2 of 3)

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

```

PennyAdmin05bSession/penny.py (Page 3 of 3)

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

from top import app

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

def _valid_username_and_password(username, password):

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

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

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

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

@app.route('/handlelogin', methods=['POST'])
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

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

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

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

PennyAdmin05bSession/auth.py (Page 2 of 2)

```

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

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

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

@app.route('/handlesignup', methods=['POST'])
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.'))

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

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)
```

PennyAdmin06SSSession/penny.py (Page 1 of 3)

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

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

import os
import dotenv
import flask
import flask_session
import flask_sqlalchemy
import database
import auth

from top import app

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

dotenv.load_dotenv()

# Session configuration to store session data in the file system
# (simple, but not realistic for a deployed app):

#app.config['SESSION_PERMANENT'] = False
#app.config['SESSION_TYPE'] = 'filesystem'
#flask_session.Session(app)

# Session configuration to store session data in a database:

session_database_url = os.getenv('SESSION_DATABASE_URL',
    'sqlite:///pennysessions.sqlite')
session_database_url = session_database_url.replace(
    'postgres://', 'postgresql://')
app.config['SESSION_PERMANENT'] = False
app.config['SESSION_TYPE'] = 'sqlalchemy'
app.config['SQLALCHEMY_DATABASE_URI'] = session_database_url
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.config['SESSION_SQLALCHEMY'] = flask_sqlalchemy.SQLAlchemy(app)
flask_session.Session(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
```

PennyAdmin06SSSession/penny.py (Page 2 of 3)

```

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

@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'])
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'
```

PennyAdmin06SSSession/penny.py (Page 3 of 3)

```
131:                 else:
132:                     message = 'The addition was successful'
133:
134:     html_code = flask.render_template('reportresults.html',
135:         username=username, message=message)
136:     response = flask.make_response(html_code)
137:     return response
138:
139: #-----------------------------------------------------------------------
140:
141: @app.route('/delete', methods=['GET'])
142: def delete():
143:
144:     auth.authenticate()
145:     username = auth.get_username()
146:
147:     if not database.is_authorized(username):
148:         html_code = 'You are not authorized to delete books.'
149:         response = flask.make_response(html_code)
150:         return response
151:
152:     html_code = flask.render_template('delete.html', username=username)
153:     response = flask.make_response(html_code)
154:     return response
155:
156: #-----------------------------------------------------------------------
157:
158: @app.route('/handledelete', methods=['POST'])
159: def handle_delete():
160:
161:     auth.authenticate()
162:     username = auth.get_username()
163:
164:     if not database.is_authorized(username):
165:         html_code = 'You are not authorized to delete books.'
166:         response = flask.make_response(html_code)
167:         return response
168:
169:     isbn = flask.request.form.get('isbn')
170:     if (isbn is None) or (isbn.strip() == ''):
171:         message = 'The deletion was unsuccessful: missing ISBN'
172:     else:
173:         isbn = isbn.strip()
174:         database.delete_book(isbn)
175:         message = 'The deletion was successful'
176:
177:     html_code = flask.render_template('reportresults.html',
178:         username=username, message=message)
179:     response = flask.make_response(html_code)
180:     return response
```

blank (Page 1 of 1)

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

csrfattack.html (Page 1 of 1)

```
<!DOCTYPE html>
<html>
   <head>
      <title>PennyAdmin</title>
   </head>
   <body>
    <hr>Hello, and welcome to <strong>PennyAdmin</strong><hr>
      <h1>Search for a Book by ISBN</h1>
      <form action="http://localhost:55555/handledelete" method="post">
         Enter an ISBN:
         <input type="text" name="isbn" autofocus>
         <input type="submit" value="Go">
      </form>
      <br>
      <br>
      <a href="http://localhost:55555/menu">Return to menu page</a>
      <br>
      <hr>
      <a href="http://localhost:55555/logout">Log out</a>
      of the application</br>
      <hr>
      Created by <a href="https://www.cs.princeton.edu/~rdondero">
      Bob Dondero</a>
      <hr>
   </body>
</html>
```

blank (Page 1 of 1)

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

PennyAdmin07aCsrfToken/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>
```

PennyAdmin07aCsrfToken/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>
```

PennyAdmin07aCsrfToken/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>
```

PennyAdmin07aCsrfToken/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>
```

PennyAdmin07aCsrfToken/penny.py (Page 1 of 4)

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

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

import os
import dotenv
import flask
import flask_session
import flask_sqlalchemy
import database
import auth

from top import app

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

dotenv.load_dotenv()

# Session configuration to store session data in the file system
# (simple, but not realistic for a deployed app):

#app.config['SESSION_PERMANENT'] = False
#app.config['SESSION_TYPE'] = 'filesystem'
#flask_session.Session(app)

# Session configuration to store session data in a database:

session_database_url = os.getenv('SESSION_DATABASE_URL',
    'sqlite:///pennysessions.sqlite')
session_database_url = session_database_url.replace(
    'postgres://', 'postgresql://')
app.config['SESSION_PERMANENT'] = False
app.config['SESSION_TYPE'] = 'sqlalchemy'
app.config['SQLALCHEMY_DATABASE_URI'] = session_database_url
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.config['SESSION_SQLALCHEMY'] = flask_sqlalchemy.SQLAlchemy(app)
flask_session.Session(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
```

PennyAdmin07aCsrfToken/penny.py (Page 2 of 4)

```

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

@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'])
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>'
        html_code += '<h1>Bad Request</h1>'
        html_code += '<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'
```

PennyAdmin07aCsrfToken/penny.py (Page 3 of 4)

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

PennyAdmin07aCsrfToken/penny.py (Page 4 of 4)

```
196:         or (csrf_token_from_request != csrf_token_from_session)):
197:         html_code = '<title>400 Bad Request</title>'
198:         html_code += '<h1>Bad Request</h1>'
199:         html_code += '<p>The CSRF token is missing or bad.</p>'
200:         response = flask.make_response(html_code)
201:         return response
202:
203:     isbn = flask.request.form.get('isbn')
204:     if (isbn is None) or (isbn.strip() == ''):
205:         message = 'The deletion was unsuccessful: missing ISBN'
206:     else:
207:         isbn = isbn.strip()
208:         database.delete_book(isbn)
209:         message = 'The deletion was successful'
210:
211:     html_code = flask.render_template('reportresults.html',
212:         username=username, message=message)
213:     response = flask.make_response(html_code)
214:     return response
```

PennyAdmin07aCsrfToken/auth.py (Page 1 of 3)

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

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

import os
import flask
import database
from top import app
#-------------------------------------------------------------------------

def _valid_username_and_password(username, password):

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

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

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

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

@app.route('/handlelogin', methods=['POST'])
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>'
        html_code += '<h1>Bad Request</h1>'
        html_code += '<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('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(
```

PennyAdmin07aCsrfToken/auth.py (Page 2 of 3)

```
            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

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

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

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

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

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

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

@app.route('/handlesignup', methods=['POST'])
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>'
        html_code += '<h1>Bad Request</h1>'
        html_code += '<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(
            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(
```

PennyAdmin07aCsrfToken/auth.py (Page 3 of 3)

```
131:         flask.url_for('login', msg='You now are signed up.'))
132:
133: #-----------------------------------------------------------------------
134:
135: def get_username():
136:
137:     return flask.session.get('username')
138:
139: #-----------------------------------------------------------------------
140:
141: def authenticate():
142:
143:     username = flask.session.get('username')
144:     if username is None:
145:         response = flask.redirect(flask.url_for('login'))
146:         flask.session['original_url'] = flask.request.url
147:         flask.abort(response)
```

blank (Page 1 of 1)

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

PennyAdmin07bCsrfToken/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>
```

PennyAdmin07bCsrfToken/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>
```

PennyAdmin07bCsrfToken/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>
```

PennyAdmin07bCsrfToken/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>
```

PennyAdmin07bCsrfToken/penny.py (Page 1 of 3)

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

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

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

from top import app

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

dotenv.load_dotenv()
app.secret_key = os.environ['APP_SECRET_KEY']
flask_wtf.csrf.CSRFProtect(app)

# Session configuration to store session data in the file system
# (simple, but not realistic for a deployed app):

#app.config['SESSION_PERMANENT'] = False
#app.config['SESSION_TYPE'] = 'filesystem'
#flask_session.Session(app)

# Session configuration to store session data in a database:

session_database_url = os.getenv('SESSION_DATABASE_URL',
    'sqlite:///pennysessions.sqlite')
session_database_url = session_database_url.replace(
    'postgres://', 'postgresql://')
app.config['SESSION_PERMANENT'] = False
app.config['SESSION_TYPE'] = 'sqlalchemy'
app.config['SQLALCHEMY_DATABASE_URI'] = session_database_url
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.config['SESSION_SQLALCHEMY'] = flask_sqlalchemy.SQLAlchemy(app)
flask_session.Session(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,
```

PennyAdmin07bCsrfToken/penny.py (Page 2 of 3)

```
        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'])
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()

```

PennyAdmin07bCsrfToken/penny.py (Page 3 of 3)

```
131:                 successful = database.add_book(isbn, author, title)
132:                 if not successful:
133:                     message = 'The addition was unsuccessful: '
134:                     message += 'duplicate ISBN'
135:                 else:
136:                     message = 'The addition was successful'
137:
138:     html_code = flask.render_template('reportresults.html',
139:         username=username, message=message)
140:     response = flask.make_response(html_code)
141:     return response
142:
143: #-----------------------------------------------------------------------
144:
145: @app.route('/delete', methods=['GET'])
146: def 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:     html_code = flask.render_template('delete.html', username=username)
157:
158:     response = flask.make_response(html_code)
159:     return response
160:
161: #-----------------------------------------------------------------------
162:
163: @app.route('/handledelete', methods=['POST'])
164: def handle_delete():
165:
166:     auth.authenticate()
167:     username = auth.get_username()
168:
169:     if not database.is_authorized(username):
170:         html_code = 'You are not authorized to delete books.'
171:         response = flask.make_response(html_code)
172:         return response
173:
174:     isbn = flask.request.form.get('isbn')
175:     if (isbn is None) or (isbn.strip() == ''):
176:         message = 'The deletion was unsuccessful: missing ISBN'
177:     else:
178:         isbn = isbn.strip()
179:         database.delete_book(isbn)
180:         message = 'The deletion was successful'
181:
182:     html_code = flask.render_template('reportresults.html',
183:         username=username, message=message)
184:     response = flask.make_response(html_code)
185:     return response
```

blank (Page 1 of 1)

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

PennyAdmin07bCsrfToken/auth.py (Page 1 of 2)

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

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

import flask
import database

from top import app

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

def _valid_username_and_password(username, password):

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

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

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

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

@app.route('/handlelogin', methods=['POST'])
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

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

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

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

PennyAdmin07bCsrfToken/auth.py (Page 2 of 2)

```

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

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

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

@app.route('/handlesignup', methods=['POST'])
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.'))

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

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)
```