PennyAdmin06Auth/penny.sql (Page 1 of 1)

```
DROP TABLE IF EXISTS books;
CREATE TABLE books (isbn TEXT PRIMARY KEY, author TEXT, title TEXT);
INSERT INTO books (isbn, author, title)
   VALUES ('123', 'Kernighan','The Practice of Programming');
INSERT INTO books (isbn, author, title)
   VALUES ('234', 'Kernighan','The C Programming Language');
INSERT INTO books (isbn, author, title)
   VALUES ('345', 'Sedgewick','Algorithms in C');

DROP TABLE IF EXISTS users;
CREATE TABLE users (username TEXT PRIMARY KEY, password TEXT);
INSERT INTO users (username, password) VALUES ('rdondero', 'xxx');
INSERT INTO users (username, password) VALUES ('bwk', 'yyy');
INSERT INTO users (username, password) VALUES ('rs', 'zzz');

DROP TABLE IF EXISTS authorizedusers;
CREATE TABLE authorizedusers (username TEXT PRIMARY KEY);
INSERT INTO authorizedusers (username) VALUES ('rdondero');
INSERT INTO authorizedusers (username) VALUES ('bwk');
```

PennyAdmin06Auth/database.py (Page 1 of 3)

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

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

import os
import sqlalchemy
import sqlalchemy.orm
import dotenv

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

dotenv.load_dotenv()
_database_url = os.getenv('DATABASE_URL', 'sqlite:///penny.sqlite')
_database_url = _database_url.replace('postgres://', 'postgresql://')

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

class Base(sqlalchemy.orm.DeclarativeBase):
    pass

class Book (Base):
    __tablename__ = 'books'
    isbn = sqlalchemy.Column(sqlalchemy.String, primary_key=True)
    author = sqlalchemy.Column(sqlalchemy.String)
    title = sqlalchemy.Column(sqlalchemy.String)

class User (Base):
    __tablename__ = 'users'
    username = sqlalchemy.Column(sqlalchemy.String, primary_key=True)
    password = sqlalchemy.Column(sqlalchemy.String)

class AuthorizedUser (Base):
    __tablename__ = 'authorizedusers'
    username = sqlalchemy.Column(sqlalchemy.String, primary_key=True)

_engine = sqlalchemy.create_engine(_database_url)

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

def get_books():

    books = []

    with sqlalchemy.orm.Session(_engine) as session:
        query = session.query(Book)
        table = query.all()
        for row in table:
            book = {'isbn': row.isbn, 'author': row.author,
                'title': row.title}
            books.append(book)

    return books

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

def add_book(isbn, author, title):

    with sqlalchemy.orm.Session(_engine) as session:
        row = Book(isbn=isbn, author=author, title=title)
        session.add(row)
        try:
            session.commit()
```

PennyAdmin06Auth/database.py (Page 2 of 3)

```
            return True
        except sqlalchemy.exc.IntegrityError:
            return False

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

def delete_book(isbn):

    with sqlalchemy.orm.Session(_engine) as session:
        session.query(Book).filter(Book.isbn==isbn).delete()
        session.commit()

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

def get_password(username):

    with sqlalchemy.orm.Session(_engine) as session:
        query = session.query(User).filter(User.username==username)
        try:
            row = query.one()
            return row.password
        except sqlalchemy.exc.NoResultFound:
            return None

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

def is_authorized(username):

    with sqlalchemy.orm.Session(_engine) as session:
        query = session.query(AuthorizedUser) \
            .filter(AuthorizedUser.username==username)
        try:
            query.one()
            return True
        except sqlalchemy.exc.NoResultFound:
            return False

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

def add_user(username, password):

    with sqlalchemy.orm.Session(_engine) as session:
        row = User(username=username, password=password)
        session.add(row)
        try:
            session.commit()
            return True
        except sqlalchemy.exc.IntegrityError:
            return False

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

# For testing:

def _write_books(books):
    for book in books:
        print('%s | %s | %s' % (book['isbn'], book['author'],
            book['title']))

def _test():
    print('-----------------------------------')
    print('Testing get_books()')
    print('-----------------------------------')
    print()
    books = get_books()
```

PennyAdmin06Auth/database.py (Page 3 of 3)

```
    _write_books(books)
    print()

    print('-----------------------------------')
    print('Testing add_book()')
    print('-----------------------------------')
    print()
    successful = add_book('456', 'Kernighan', 'New Book')
    if successful:
        print('Add was successful')
        print()
        books = get_books()
        _write_books(books)
        print()
    else:
        print('Add was unsuccessful')
        print()
        _write_books(books)
        print()
    successful = add_book('456', 'Kernighan', 'New Book')
    if successful:
        print('Add was successful')
        print()
        books = get_books()
        _write_books(books)
        print()
    else:
        print('Add was unsuccessful')
        print()
        _write_books(books)
        print()

    print('-----------------------------------')
    print('Testing delete_book()')
    print('-----------------------------------')
    print()
    delete_book('456')
    books = get_books()
    _write_books(books)
    print()
    delete_book('456')
    books = get_books()
    _write_books(books)
    print()

    print('-----------------------------------')
    print('Testing get_password()')
    print('-----------------------------------')
    print()
    password = get_password('rdondero')
    print(password)
    password = get_password('rdondero2')
    print(password)
    print()

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

PennyAdmin06Auth/header.html (Page 1 of 1)

```
1: <hr>Hello {{username}}, and welcome to <strong>PennyAdmin</strong><hr>
```

PennyAdmin06Auth/footer.html (Page 1 of 1)

```
1: <hr>
2: <a href="logout">Log out</a> of the application</br>
3: <hr>
4: Created by <a href="https://www.cs.princeton.edu/~rdondero">
5: Bob Dondero</a>
6: <hr>
```

PennyAdmin06Auth/index.html (Page 1 of 1)

```
<!DOCTYPE html>
<html>
   <head>
      <title>PennyAdmin</title>
   </head>
   <body>
      {% include 'header.html' %}
      <br>
      <a href="/show">Show all books</a><br>
      {% if is_authorized: %}
         <a href="/add">Add a book</a><br>
         <a href="/delete">Delete a book by ISBN</a><br>
      {% endif %}
      <br>
      {% include 'footer.html' %}
   </body>
</html>
```

PennyAdmin06Auth/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">
         <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>
```

PennyAdmin06Auth/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">
         <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>
```

PennyAdmin06Auth/loggedout.html (Page 1 of 1)

```
<!DOCTYPE html>
<html>
   <head>
      <title>PennyAdmin</title>
   </head>
   <body>
      <h2>You are logged out of PennyAdmin</h2>
      <a href="/index">Log back in</a>
   </body>
</html>
```

PennyAdmin06Auth/top.py (Page 1 of 1)

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

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

import flask

app = flask.Flask('penny', template_folder='.')
```

PennyAdmin06Auth/penny.py (Page 1 of 3)

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

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

import flask
import database
import auth

from top import app

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

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

    username = auth.authenticate()
    is_authorized = database.is_authorized(username)

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

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

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

    username = auth.authenticate()

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

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

def report_results(username, message1, message2):

    html_code = flask.render_template('reportresults.html',
        username=username, message1=message1, message2=message2)
    response = flask.make_response(html_code)
    return response

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

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

    username = auth.authenticate()
    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

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

PennyAdmin06Auth/penny.py (Page 2 of 3)

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

PennyAdmin06Auth/penny.py (Page 3 of 3)

```
131:         return report_results(username, 'Missing ISBN', '')
132:
133:     isbn = isbn.strip()
134:
135:     database.delete_book(isbn)
136:
137:     message1 = 'The deletion was successful'
138:     message2 = 'The database now does not contain a book with ISBN '
139:     message2 += isbn
140:
141:     return report_results(username, message1, message2)
```

PennyAdmin06Auth/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.request.cookies.get('original_url', '/index')
    response = flask.redirect(original_url)
    response.set_cookie('username', username)
    return response

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

@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.
```

PennyAdmin06Auth/auth.py (Page 2 of 2)

```
    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 authenticate():

    username = flask.request.cookies.get('username')
    if username is None:
        response = flask.redirect(flask.url_for('login'))
        response.set_cookie('original_url', flask.request.url)
        flask.abort(response)
    return username
```

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))

            out_flo = sock.makefile(mode='w', encoding='iso-8859-1')
            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')
            out_flo.flush()

            in_flo = sock.makefile(mode='r', encoding='iso-8859-1')
            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.
```

PennyAdmin07Encrypt/auth.py (Page 1 of 2)

```
#!/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

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

```

PennyAdmin07Encrypt/auth.py (Page 2 of 2)

```
@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 authenticate():

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

    if encrypted_username is None:
        response = flask.redirect(flask.url_for('login'))
        response.set_cookie('original_url', flask.request.url)
        flask.abort(response)

    username = cryptocode.decrypt(encrypted_username, secret_key)
    if not username:
        response = flask.redirect(flask.url_for('login'))
        response.set_cookie('original_url', flask.request.url)
        flask.abort(response)

    return username
```

PennyAdmin08Session/top.py (Page 1 of 1)

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

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

import os
import dotenv
import flask

app = flask.Flask('penny', template_folder='.')

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

blank (Page 1 of 1)

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

PennyAdmin08Session/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)
```

PennyAdmin08Session/auth.py (Page 2 of 2)

```
    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 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)
    return username
```