

PennyAdmin06Auth/penny.sql (Page 1 of 1)

```

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

```

PennyAdmin06Auth/database.py (Page 1 of 3)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # database.py
5: # Author: Bob Dondero
6: #-----
7:
8: import os
9: import sqlalchemy
10: import sqlalchemy.orm
11: import dotenv
12:
13: #-----
14:
15: dotenv.load_dotenv()
16: _database_url = os.getenv('DATABASE_URL', 'sqlite:///penny.sqlite')
17: _database_url = _database_url.replace('postgres://', 'postgresql://')
18:
19: #-----
20:
21: class Base(sqlalchemy.orm.DeclarativeBase):
22:     pass
23:
24: class Book(Base):
25:     __tablename__ = 'books'
26:     isbn = sqlalchemy.Column(sqlalchemy.String, primary_key=True)
27:     author = sqlalchemy.Column(sqlalchemy.String)
28:     title = sqlalchemy.Column(sqlalchemy.String)
29:
30: class User(Base):
31:     __tablename__ = 'users'
32:     username = sqlalchemy.Column(sqlalchemy.String, primary_key=True)
33:     password = sqlalchemy.Column(sqlalchemy.String)
34:
35: class AuthorizedUser(Base):
36:     __tablename__ = 'authorizedusers'
37:     username = sqlalchemy.Column(sqlalchemy.String, primary_key=True)
38:
39: _engine = sqlalchemy.create_engine(_database_url)
40:
41: #-----
42:
43: def get_books():
44:
45:     books = []
46:
47:     with sqlalchemy.orm.Session(_engine) as session:
48:         query = session.query(Book)
49:         table = query.all()
50:         for row in table:
51:             book = {'isbn': row.isbn, 'author': row.author,
52:                   'title': row.title}
53:             books.append(book)
54:
55:     return books
56:
57: #-----
58:
59: def add_book(isbn, author, title):
60:
61:     with sqlalchemy.orm.Session(_engine) as session:
62:         row = Book(isbn=isbn, author=author, title=title)
63:         session.add(row)
64:         try:
65:             session.commit()

```

PennyAdmin06Auth/database.py (Page 2 of 3)

```

66:         return True
67:     except sqlalchemy.exc.IntegrityError:
68:         return False
69:
70: #-----
71:
72: def delete_book(isbn):
73:
74:     with sqlalchemy.orm.Session(_engine) as session:
75:         session.query(Book).filter(Book.isbn==isbn).delete()
76:         session.commit()
77:
78: #-----
79:
80: def get_password(username):
81:
82:     with sqlalchemy.orm.Session(_engine) as session:
83:         query = session.query(User).filter(User.username==username)
84:         try:
85:             row = query.one()
86:             return row.password
87:         except sqlalchemy.exc.NoResultFound:
88:             return None
89:
90: #-----
91:
92: def is_authorized(username):
93:
94:     with sqlalchemy.orm.Session(_engine) as session:
95:         query = session.query(AuthorizedUser) \
96:             .filter(AuthorizedUser.username==username)
97:         try:
98:             query.one()
99:             return True
100:        except sqlalchemy.exc.NoResultFound:
101:            return False
102:
103: #-----
104:
105: def add_user(username, password):
106:
107:     with sqlalchemy.orm.Session(_engine) as session:
108:         row = User(username=username, password=password)
109:         session.add(row)
110:         try:
111:             session.commit()
112:             return True
113:         except sqlalchemy.exc.IntegrityError:
114:             return False
115:
116: #-----
117:
118: # For testing:
119:
120: def _write_books(books):
121:     for book in books:
122:         print('%s | %s | %s' % (book['isbn'], book['author'],
123:             book['title']))
124:
125: def _test():
126:     print('-----')
127:     print('Testing get_books()')
128:     print('-----')
129:     print()
130:     books = get_books()

```

PennyAdmin06Auth/database.py (Page 3 of 3)

```

131:     _write_books(books)
132:     print()
133:
134:     print('-----')
135:     print('Testing add_book()')
136:     print('-----')
137:     print()
138:     successful = add_book('456', 'Kernighan', 'New Book')
139:     if successful:
140:         print('Add was successful')
141:         print()
142:         books = get_books()
143:         _write_books(books)
144:         print()
145:     else:
146:         print('Add was unsuccessful')
147:         print()
148:         _write_books(books)
149:         print()
150:     successful = add_book('456', 'Kernighan', 'New Book')
151:     if successful:
152:         print('Add was successful')
153:         print()
154:         books = get_books()
155:         _write_books(books)
156:         print()
157:     else:
158:         print('Add was unsuccessful')
159:         print()
160:         _write_books(books)
161:         print()
162:
163:     print('-----')
164:     print('Testing delete_book()')
165:     print('-----')
166:     print()
167:     delete_book('456')
168:     books = get_books()
169:     _write_books(books)
170:     print()
171:     delete_book('456')
172:     books = get_books()
173:     _write_books(books)
174:     print()
175:
176:     print('-----')
177:     print('Testing get_password()')
178:     print('-----')
179:     print()
180:     password = get_password('rdontero')
181:     print(password)
182:     password = get_password('rdontero2')
183:     print(password)
184:     print()
185:
186: if __name__ == '__main__':
187:     _test()

```

PennyAdmin06Auth/header.html (Page 1 of 1)

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

PennyAdmin06Auth/footer.html (Page 1 of 1)

1: `<hr>`
2: `Log out of the application</br>`
3: `<hr>`
4: `Created by `
5: `Bob Dondero`
6: `<hr>`

PennyAdmin06Auth/index.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>PennyAdmin</title>
5:   </head>
6:   <body>
7:     {% include 'header.html' %}
8:     <br>
9:     <a href="/show">Show all books</a><br>
10:    {% if is_authorized: %}
11:      <a href="/add">Add a book</a><br>
12:      <a href="/delete">Delete a book by ISBN</a><br>
13:    {% endif %}
14:    <br>
15:    {% include 'footer.html' %}
16:  </body>
17: </html>

```

PennyAdmin06Auth/login.html (Page 1 of 1)

```

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

```

PennyAdmin06Auth/signup.html (Page 1 of 1)

```

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

```

PennyAdmin06Auth/loggedout.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>PennyAdmin</title>
5:   </head>
6:   <body>
7:     <h2>You are logged out of PennyAdmin</h2>
8:     <a href="/index">Log back in</a>
9:   </body>
10: </html>

```

PennyAdmin06Auth/top.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # top.py
5: # Author: Bob Dondero
6: #-----
7:
8: import flask
9:
10: app = flask.Flask('penny', template_folder='.')
```

PennyAdmin06Auth/penny.py (Page 1 of 3)

```

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

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)

```

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

```

PennyAdmin06Auth/auth.py (Page 2 of 2)

```

66:     response.set_cookie('username', '', expires=0)
67:     response.set_cookie('original_url', '', expires=0)
68:
69:     return response
70:
71: #-----
72:
73: @app.route('/signup', methods=['GET'])
74: def signup():
75:
76:     error_msg = flask.request.args.get('error_msg')
77:     if error_msg is None:
78:         error_msg = ''
79:
80:     html_code = flask.render_template('signup.html',
81:                                       error_msg=error_msg)
82:     response = flask.make_response(html_code)
83:     return response
84:
85: #-----
86:
87: @app.route('/handlesignup', methods=['POST'])
88: def handle_signup():
89:
90:     username = flask.request.form.get('username')
91:     password = flask.request.form.get('password')
92:     if (username is None) or (username.strip() == ''):
93:         return flask.redirect(
94:             flask.url_for('signup', error_msg='Invalid username'))
95:     if (password is None) or (password.strip() == ''):
96:         return flask.redirect(
97:             flask.url_for('signup', error_msg='Invalid password'))
98:     successful = database.add_user(username, password)
99:     if not successful:
100:         return flask.redirect(
101:             flask.url_for('signup', error_msg='Duplicate username'))
102:
103:     return flask.redirect(
104:         flask.url_for('login', msg='You now are signed up.))
105:
106: #-----
107:
108: def authenticate():
109:
110:     username = flask.request.cookies.get('username')
111:     if username is None:
112:         response = flask.redirect(flask.url_for('login'))
113:         response.set_cookie('original_url', flask.request.url)
114:         flask.abort(response)
115:     return username
116:

```


cookieforgeryattack.py (Page 1 of 1)

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

blank (Page 1 of 1)

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

PennyAdmin07Encrypt/auth.py (Page 1 of 2)

```

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

```

PennyAdmin07Encrypt/auth.py (Page 2 of 2)

```

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

```

PennyAdmin08Session/top.py (Page 1 of 1)

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

blank (Page 1 of 1)

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

PennyAdmin08Session/auth.py (Page 1 of 2)

```

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

```

PennyAdmin08Session/auth.py (Page 2 of 2)

```

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

```