

PennyAdmin04Auth/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');

```

PennyAdmin04Auth/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: _DATABASE_URL = _DATABASE_URL.replace(
19:     'postgresql://', 'postgresql+psycopg://')
20:
21: #-----
22:
23: class Base(sqlalchemy.orm.DeclarativeBase):
24:     pass
25:
26: class Book(Base):
27:     __tablename__ = 'books'
28:     isbn = sqlalchemy.Column(sqlalchemy.String, primary_key=True)
29:     author = sqlalchemy.Column(sqlalchemy.String)
30:     title = sqlalchemy.Column(sqlalchemy.String)
31:
32: class User(Base):
33:     __tablename__ = 'users'
34:     username = sqlalchemy.Column(sqlalchemy.String, primary_key=True)
35:     password = sqlalchemy.Column(sqlalchemy.String)
36:
37: class AuthorizedUser(Base):
38:     __tablename__ = 'authorizedusers'
39:     username = sqlalchemy.Column(sqlalchemy.String, primary_key=True)
40:
41: _engine = sqlalchemy.create_engine(_DATABASE_URL)
42:
43: #-----
44:
45: def get_books():
46:
47:     books = []
48:
49:     with sqlalchemy.orm.Session(_engine) as session:
50:         query = session.query(Book)
51:         table = query.all()
52:         for row in table:
53:             book = {'isbn': row.isbn, 'author': row.author,
54:                   'title': row.title}
55:             books.append(book)
56:
57:     return books
58:
59: #-----
60:
61: def add_book(isbn, author, title):
62:
63:     with sqlalchemy.orm.Session(_engine) as session:
64:         row = Book(isbn=isbn, author=author, title=title)
65:         session.add(row)

```

PennyAdmin04Auth/database.py (Page 2 of 3)

```

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

```

PennyAdmin04Auth/database.py (Page 3 of 3)

```

131:     books = get_books()
132:     _write_books(books)
133:     print()
134:
135:     print('-----')
136:     print('Testing add_book()')
137:     print('-----')
138:     print()
139:     successful = add_book('456', 'Kernighan', 'New Book')
140:     if successful:
141:         print('Add was successful')
142:         print()
143:         books = get_books()
144:         _write_books(books)
145:         print()
146:     else:
147:         print('Add was unsuccessful')
148:         print()
149:         _write_books(books)
150:         print()
151:     successful = add_book('456', 'Kernighan', 'New Book')
152:     if successful:
153:         print('Add was successful')
154:         print()
155:         books = get_books()
156:         _write_books(books)
157:         print()
158:     else:
159:         print('Add was unsuccessful')
160:         print()
161:         _write_books(books)
162:         print()
163:
164:     print('-----')
165:     print('Testing delete_book()')
166:     print('-----')
167:     print()
168:     delete_book('456')
169:     books = get_books()
170:     _write_books(books)
171:     print()
172:     delete_book('456')
173:     books = get_books()
174:     _write_books(books)
175:     print()
176:
177:     print('-----')
178:     print('Testing get_password()')
179:     print('-----')
180:     print()
181:     password = get_password('rdonero')
182:     print(password)
183:     password = get_password('rdonero2')
184:     print(password)
185:     print()
186:
187: if __name__ == '__main__':
188:     _test()

```

PennyAdmin04Auth/header.html (Page 1 of 1)

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

PennyAdmin04Auth/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/~rdonero">
5: Bob Dondero</a>
6: <hr>
```

PennyAdmin04Auth/menu.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>

```

PennyAdmin04Auth/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>

```

PennyAdmin04Auth/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>

```

PennyAdmin04Auth/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">Revisit the application</a>
9:   </body>
10: </html>

```

PennyAdmin04Auth/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: #-----
13:
14: app = flask.Flask(__name__, template_folder='.')
15:
16: auth.init(app)
17:
18: #-----
19:
20: @app.route('/', methods=['GET'])
21: @app.route('/index', methods=['GET'])
22: def index():
23:
24:     html_code = flask.render_template('index.html')
25:     response = flask.make_response(html_code)
26:     return response
27:
28: #-----
29:
30: @app.route('/menu', methods=['GET'])
31: def menu():
32:
33:     auth.authenticate()
34:     username = auth.get_username()
35:
36:     is_authorized = database.is_authorized(username)
37:
38:     html_code = flask.render_template('menu.html', username=username,
39:                                     is_authorized=is_authorized)
40:     response = flask.make_response(html_code)
41:     return response
42:
43: #-----
44:
45: @app.route('/show', methods=['GET'])
46: def show():
47:
48:     auth.authenticate()
49:     username = auth.get_username()
50:
51:     books = database.get_books()
52:     html_code = flask.render_template('show.html',
53:                                     username=username, books=books)
54:     response = flask.make_response(html_code)
55:     return response
56:
57: #-----
58:
59: @app.route('/add', methods=['GET'])
60: def add():
61:
62:     auth.authenticate()
63:     username = auth.get_username()
64:
65:     if not database.is_authorized(username):

```

PennyAdmin04Auth/penny.py (Page 2 of 3)

```

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

```

PennyAdmin04Auth/penny.py (Page 3 of 3)

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

blank (Page 1 of 1)

1: This page is intentionally blank.

PennyAdmin04Auth/auth.py (Page 1 of 2)

```

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

```

PennyAdmin04Auth/auth.py (Page 2 of 2)

```

66:     # Delete cookies in the browser by setting them to expire at
67:     # a time that is in the past.
68:     response.set_cookie('username', '', expires=0)
69:     response.set_cookie('original_url', '', expires=0)
70:
71:     return response
72:
73: #-----
74:
75: def signup():
76:
77:     error_msg = flask.request.args.get('error_msg')
78:     if error_msg is None:
79:         error_msg = ''
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: def handle_signup():
88:
89:     username = flask.request.form.get('username')
90:     password = flask.request.form.get('password')
91:     if (username is None) or (username.strip() == ''):
92:         return flask.redirect(
93:             flask.url_for('signup', error_msg='Invalid username'))
94:     if (password is None) or (password.strip() == ''):
95:         return flask.redirect(
96:             flask.url_for('signup', error_msg='Invalid password'))
97:     successful = database.add_user(username, password)
98:     if not successful:
99:         return flask.redirect(
100:             flask.url_for('signup', error_msg='Duplicate username'))
101:     return flask.redirect(
102:         flask.url_for('login', msg='You now are signed up.'))
103:
104: #-----
105: # Authentication functions
106: #-----
107:
108: def _valid_username_and_password(username, password):
109:
110:     stored_password = database.get_password(username)
111:     if stored_password is None:
112:         return False
113:     return password == stored_password
114:
115: #-----
116:
117: def get_username():
118:
119:     return flask.request.cookies.get('username')
120:
121: #-----
122:
123: def authenticate():
124:
125:     username = flask.request.cookies.get('username')
126:     if username is None:
127:         response = flask.redirect(flask.url_for('login'))
128:         response.set_cookie('original_url', flask.request.url)
129:         flask.abort(response)

```

cookieforgeryattack.py (Page 1 of 1)

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

blank (Page 1 of 1)

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

PennyAdmin05aEncrypt/auth.py (Page 1 of 3)

```

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

```

PennyAdmin05aEncrypt/auth.py (Page 2 of 3)

```

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

```

PennyAdmin05aEncrypt/auth.py (Page 3 of 3)

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

blank (Page 1 of 1)

1: This page is intentionally blank.

PennyAdmin05bSession/penny.py (Page 1 of 3)

```

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

```

PennyAdmin05bSession/penny.py (Page 2 of 3)

```

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

```

PennyAdmin05bSession/penny.py (Page 3 of 3)

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

blank (Page 1 of 1)

1: This page is intentionally blank.

PennyAdmin05bSession/auth.py (Page 1 of 2)

```

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

```

PennyAdmin05bSession/auth.py (Page 2 of 2)

```

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

```