

**PennyAdmin07aHash/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 ('rdontero',
13:   'cd2eb0837c9b4c962c22d2ff8b5441b7b45805887f051d39bf133b583baf6860');
14: INSERT INTO users (username, password) VALUES ('bwk',
15:   'f2afd1cacb5441a5e65a7a460a5f9898b7b98b08aa6323a2e53c8b9a9686cd86');
16: INSERT INTO users (username, password) VALUES ('rs',
17:   '17f165d5a5ba695f27c023a83aa2b3463e23810e360b7517127e90161eebabda');
18:
19: DROP TABLE IF EXISTS authorizedusers;
20: CREATE TABLE authorizedusers (username TEXT PRIMARY KEY);
21: INSERT INTO authorizedusers (username) VALUES ('rdontero');
22: INSERT INTO authorizedusers (username) VALUES ('bwk');
```

**blank (Page 1 of 1)**

1: This page is intentionally blank.

## PennyAdmin07aHash/auth.py (Page 1 of 3)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # auth.py
5: # Author: Bob Dondero
6: #-----
7:
8: import hashlib
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:
37:     html = flask.render_template('login.html', msg=msg)
38:
39:     response = flask.make_response(html)
40:     return response
41:
42: #-----
43:
44: def handle_login():
45:
46:     username = flask.request.form.get('username')
47:     password = flask.request.form.get('password')
48:     if (username is None) or (username.strip() == ''):
49:         return flask.redirect(
50:             flask.url_for('login', msg='Wrong username or password'))
51:     if (password is None) or (password.strip() == ''):
52:         return flask.redirect(
53:             flask.url_for('login', msg='Wrong username or password'))
54:     if not _valid_username_and_password(username, password):
55:         return flask.redirect(
56:             flask.url_for('login', msg='Wrong username or password'))
57:     original_url = flask.session.get('original_url', '/index')
58:     response = flask.redirect(original_url)
59:     flask.session['username'] = username
60:     return response
61:
62: #-----
63:
64: def logout():
65:

```

## PennyAdmin07aHash/auth.py (Page 2 of 3)

```

66:     flask.session.clear()
67:     html_code = flask.render_template('loggedout.html')
68:     response = flask.make_response(html_code)
69:     return response
70:
71: #-----
72:
73: def signup():
74:
75:     error_msg = flask.request.args.get('error_msg')
76:     if error_msg is None:
77:         error_msg = ''
78:
79:     html_code = flask.render_template('signup.html',
80:                                     error_msg=error_msg)
81:
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:
98:     hash_code = _hash(password)
99:     successful = database.add_user(username, hash_code)
100:    if not successful:
101:        return flask.redirect(
102:            flask.url_for('signup', error_msg='Duplicate username'))
103:    return flask.redirect(
104:        flask.url_for('login', msg='You now are signed up.))
105:
106: #-----
107: # Authentication functions
108: #-----
109:
110: def _hash(password):
111:
112:     hash_code = hashlib.sha256()
113:     hash_code.update(password.encode('ascii'))
114:     hash_code = hash_code.hexdigest()
115:     return hash_code
116:
117: #-----
118:
119: def _valid_username_and_password(username, password):
120:
121:     stored_hash_code = database.get_password(username)
122:     if stored_hash_code is None:
123:         return False
124:     hash_code = _hash(password)
125:     return hash_code == stored_hash_code
126:
127: #-----
128:
129: def get_username():
130:

```

**PennyAdmin07aHash/auth.py (Page 3 of 3)**

```
131:     return flask.session.get('username')
132:
133: #-----
134:
135: def authenticate():
136:
137:     username = flask.session.get('username')
138:     if username is None:
139:         response = flask.redirect(flask.url_for('login'))
140:         flask.session['original_url'] = flask.request.url
141:         flask.abort(response)
```

**blank (Page 1 of 1)**

1: This page is intentionally blank.

**PennyAdmin07bSaltHash/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',
13:   'pbkdf2:sha256:600000$UOVCFeB4bjAW4nYx$0641776b12a4054fe5fb72eb4f9bbf
73c4266099de5ec01f09c325ed56746c3a');
14: INSERT INTO users (username, password) VALUES ('bwk',
15:   'pbkdf2:sha256:600000$fQwzUw8ZCPET43r$2390d394f64f239e5bc765f4163d49
d40b97601fae4d6ae0830c52296438e6ae');
16: INSERT INTO users (username, password) VALUES ('rs',
17:   'pbkdf2:sha256:600000$xbvn5wi1R8eJ5Mq$5bb39b39b45d65628091f4ba5fabda
99b30e9a60c447818db626f35f1dffd9f');
18:
19: DROP TABLE IF EXISTS authorizedusers;
20: CREATE TABLE authorizedusers (username TEXT PRIMARY KEY);
21: INSERT INTO authorizedusers (username) VALUES ('rdonero');
22: INSERT INTO authorizedusers (username) VALUES ('bwk');
```

**blank (Page 1 of 1)**

1: This page is intentionally blank.

## PennyAdmin07bSaltHash/auth.py (Page 1 of 3)

```

1:#!/usr/bin/env python
2:
3: #-----
4: # auth.py
5: # Author: Bob Dondero
6: #-----
7:
8: import werkzeug.security
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:
37:     html = flask.render_template('login.html', msg=msg)
38:
39:     response = flask.make_response(html)
40:     return response
41:
42: #-----
43:
44: def handle_login():
45:
46:     username = flask.request.form.get('username')
47:     password = flask.request.form.get('password')
48:     if (username is None) or (username.strip() == ''):
49:         return flask.redirect(
50:             flask.url_for('login', msg='Wrong username or password'))
51:     if (password is None) or (password.strip() == ''):
52:         return flask.redirect(
53:             flask.url_for('login', msg='Wrong username or password'))
54:     if not _valid_username_and_password(username, password):
55:         return flask.redirect(
56:             flask.url_for('login', msg='Wrong username or password'))
57:     original_url = flask.session.get('original_url', '/index')
58:     response = flask.redirect(original_url)
59:     flask.session['username'] = username
60:     return response
61:
62: #-----
63:
64: def logout():
65:

```

## PennyAdmin07bSaltHash/auth.py (Page 2 of 3)

```

66:     flask.session.clear()
67:     html_code = flask.render_template('loggedout.html')
68:     response = flask.make_response(html_code)
69:     return response
70:
71: #-----
72:
73: def signup():
74:
75:     error_msg = flask.request.args.get('error_msg')
76:     if error_msg is None:
77:         error_msg = ''
78:
79:     html_code = flask.render_template('signup.html',
80:         error_msg=error_msg)
81:
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:
98:     hash_code = werkzeug.security.generate_password_hash(
99:         password, 'pbkdf2')
100:     successful = database.add_user(username, hash_code)
101:     if not successful:
102:         return flask.redirect(
103:             flask.url_for('signup', error_msg='Duplicate username'))
104:
105:     return flask.redirect(
106:         flask.url_for('login', msg='You now are signed up.))
107:
108: #-----
109: # Authentication functions
110: #-----
111:
112: def _valid_username_and_password(username, password):
113:
114:     stored_hash_code = database.get_password(username)
115:     if stored_hash_code is None:
116:         return False
117:     return werkzeug.security.check_password_hash(
118:         stored_hash_code, password)
119:
120: #-----
121:
122: def get_username():
123:
124:     return flask.session.get('username')
125:
126: #-----
127:
128: def authenticate():
129:
130:     username = flask.session.get('username')

```

**PennyAdmin07bSaltHash/auth.py (Page 3 of 3)**

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

**blank (Page 1 of 1)**

1: This page is intentionally blank.

## PennyAdmin08aHttps/penny.py (Page 1 of 3)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import os
9: import dotenv
10: import flask
11: import flask_wtf.csrf
12: import database
13: import auth
14:
15: #-----
16:
17: dotenv.load_dotenv()
18: _APP_SECRET_KEY = os.environ['APP_SECRET_KEY']
19:
20: #-----
21:
22: app = flask.Flask(__name__, template_folder='.')
23: app.secret_key = _APP_SECRET_KEY
24: auth.init(app)
25: flask_wtf.csrf.CSRFProtect(app)
26:
27: #-----
28:
29: @app.before_request
30: def before_request():
31:     is_running_locally = 'localhost:' in flask.request.url_root
32:     is_using_https = flask.request.is_secure
33:     if (not is_running_locally) and (not is_using_https):
34:         url = flask.request.url.replace('http://', 'https://', 1)
35:         return flask.redirect(url, code=301)
36:     return None
37:
38: #-----
39:
40: @app.route('/', methods=['GET'])
41: @app.route('/index', methods=['GET'])
42: def index():
43:
44:     html_code = flask.render_template('index.html')
45:     response = flask.make_response(html_code)
46:     return response
47:
48: #-----
49:
50: @app.route('/menu', methods=['GET'])
51: def menu():
52:
53:     auth.authenticate()
54:     username = auth.get_username()
55:
56:     is_authorized = database.is_authorized(username)
57:
58:     html_code = flask.render_template('menu.html', username=username,
59:                                     is_authorized=is_authorized)
60:     response = flask.make_response(html_code)
61:     return response
62:
63: #-----
64:
65: @app.route('/show', methods=['GET'])

```

## PennyAdmin08aHttps/penny.py (Page 2 of 3)

```

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

```

## PennyAdmin08aHttps/penny.py (Page 3 of 3)

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

## blank (Page 1 of 1)

1: This page is intentionally blank.

## PennyAdmin08bHttps/penny.py (Page 1 of 3)

```

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

```

## PennyAdmin08bHttps/penny.py (Page 2 of 3)

```

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

```

## PennyAdmin08bHttps/penny.py (Page 3 of 3)

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

## blank (Page 1 of 1)

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

## PennyAdmin08cHttpsLocal/runserver.py (Page 1 of 1)

```
1: #!/usr/bin/env python
2:
3: #-----
4: # runserver.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import penny
10:
11: def main():
12:
13:     if len(sys.argv) != 2:
14:         print(f'usage: {sys.argv[0]} port', file=sys.stderr)
15:         sys.exit(1)
16:
17:     try:
18:         port = int(sys.argv[1])
19:     except Exception:
20:         print(f'{sys.argv[0]}: Port must be an integer.',
21:               file=sys.stderr)
22:         sys.exit(1)
23:
24:     try:
25:         penny.app.run(host='0.0.0.0', port=port, debug=True,
26:                       ssl_context=('cert.pem', 'key.pem'))
27:     except Exception as ex:
28:         print(f'{sys.argv[0]}: {ex}', file=sys.stderr)
29:         sys.exit(1)
30:
31: if __name__ == '__main__':
32:     main()
```

## blank (Page 1 of 1)

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