

PennyAdmin08aHash/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.

PennyAdmin08aHash/auth.py (Page 1 of 2)

```

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: from top import app
13:
14: #-----
15:
16: def _hash(password):
17:
18:     hash_code = hashlib.sha256()
19:     hash_code.update(password.encode('ascii'))
20:     hash_code = hash_code.hexdigest()
21:     return hash_code
22:
23: #-----
24:
25: def _valid_username_and_password(username, password):
26:
27:     stored_hash_code = database.get_password(username)
28:     if stored_hash_code is None:
29:         return False
30:     hash_code = _hash(password)
31:     return hash_code == stored_hash_code
32:
33: #-----
34:
35: @app.route('/login', methods=['GET'])
36: def login():
37:
38:     msg = flask.request.args.get('msg')
39:     if msg is None:
40:         msg = ''
41:
42:     html = flask.render_template('login.html', msg=msg)
43:
44:     response = flask.make_response(html)
45:     return response
46:
47: #-----
48:
49: @app.route('/handlelogin', methods=['POST'])
50: def handle_login():
51:
52:     username = flask.request.form.get('username')
53:     password = flask.request.form.get('password')
54:     if (username is None) or (username.strip() == ''):
55:         return flask.redirect(
56:             flask.url_for('login', msg='Wrong username or password'))
57:     if (password is None) or (password.strip() == ''):
58:         return flask.redirect(
59:             flask.url_for('login', msg='Wrong username or password'))
60:     if not _valid_username_and_password(username, password):
61:         return flask.redirect(
62:             flask.url_for('login', msg='Wrong username or password'))
63:     original_url = flask.session.get('original_url', '/index')
64:     response = flask.redirect(original_url)
65:     flask.session['username'] = username

```

PennyAdmin08aHash/auth.py (Page 2 of 2)

```

66:     return response
67:
68: #-----
69:
70: @app.route('/logout', methods=['GET'])
71: def logout():
72:
73:     flask.session.clear()
74:     html_code = flask.render_template('loggedout.html')
75:     response = flask.make_response(html_code)
76:     return response
77:
78: #-----
79:
80: @app.route('/signup', methods=['GET'])
81: def signup():
82:
83:     error_msg = flask.request.args.get('error_msg')
84:     if error_msg is None:
85:         error_msg = ''
86:
87:     html_code = flask.render_template('signup.html',
88:         error_msg=error_msg)
89:
90:     response = flask.make_response(html_code)
91:     return response
92:
93: #-----
94:
95: @app.route('/handlesignup', methods=['POST'])
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:
107:    hash_code = _hash(password)
108:    successful = database.add_user(username, hash_code)
109:    if not successful:
110:        return flask.redirect(
111:            flask.url_for('signup', error_msg='Duplicate username'))
112:
113:    return flask.redirect(
114:        flask.url_for('login', msg='You now are signed up.'))
115:
116: #-----
117:
118: def get_username():
119:
120:     return flask.session.get('username')
121:
122: #-----
123:
124: def authenticate():
125:
126:     username = flask.session.get('username')
127:     if username is None:
128:         response = flask.redirect(flask.url_for('login'))
129:         flask.session['original_url'] = flask.request.url
130:         flask.abort(response)

```

PennyAdmin08bSaltHash/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$2390d394f64f239e5bc765f4163d49
d40b97601fae4d6ae0830c52296438e6ae');
16: INSERT INTO users (username, password) VALUES ('rs',
17:   'pbkdf2:sha256:600000$xbvn5wijlR8eJ5Mq$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.

PennyAdmin08bSaltHash/auth.py (Page 1 of 2)

```

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: from top import app
13:
14: #-----
15:
16: def _valid_username_and_password(username, password):
17:
18:     stored_hash_code = database.get_password(username)
19:     if stored_hash_code is None:
20:         return False
21:     return werkzeug.security.check_password_hash(
22:         stored_hash_code, 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:
35:     response = flask.make_response(html)
36:     return response
37:
38: #-----
39:
40: @app.route('/handlelogin', methods=['POST'])
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.session.get('original_url', '/index')
55:     response = flask.redirect(original_url)
56:     flask.session['username'] = username
57:     return response
58:
59: #-----
60:
61: @app.route('/logout', methods=['GET'])
62: def logout():
63:
64:     flask.session.clear()
65:     html_code = flask.render_template('loggedout.html')

```

PennyAdmin08bSaltHash/auth.py (Page 2 of 2)

```

66:     response = flask.make_response(html_code)
67:     return response
68:
69: #-----
70:
71: @app.route('/signup', methods=['GET'])
72: def signup():
73:
74:     error_msg = flask.request.args.get('error_msg')
75:     if error_msg is None:
76:         error_msg = ''
77:
78:     html_code = flask.render_template('signup.html',
79:         error_msg=error_msg)
80:
81:     response = flask.make_response(html_code)
82:     return response
83:
84: #-----
85:
86: @app.route('/handlesignup', methods=['POST'])
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:
110: def get_username():
111:
112:     return flask.session.get('username')
113:
114: #-----
115:
116: def authenticate():
117:
118:     username = flask.session.get('username')
119:     if username is None:
120:         response = flask.redirect(flask.url_for('login'))
121:         flask.session['original_url'] = flask.request.url
122:         flask.abort(response)

```

PennyAdmin09aHttps/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_session
12: import flask_sqlalchemy
13: import flask_wtf.csrf
14: import database
15: import auth
16: from top import app
17:
18: #-----
19:
20: dotenv.load_dotenv()
21: app.secret_key = os.environ['APP_SECRET_KEY']
22: flask_wtf.csrf.CSRFProtect(app)
23:
24: # Session configuration to store session data in the file system
25: # (simple, but not realistic for a deployed app):
26:
27: #app.config['SESSION_PERMANENT'] = False
28: #app.config['SESSION_TYPE'] = 'filesystem'
29: #flask_session.Session(app)
30:
31: # Session configuration to store session data in a database:
32:
33: session_database_url = os.getenv('SESSION_DATABASE_URL',
34:     'sqlite:///pennysessions.sqlite')
35: session_database_url = session_database_url.replace(
36:     'postgres://', 'postgresql://')
37: app.config['SESSION_PERMANENT'] = False
38: app.config['SESSION_TYPE'] = 'sqlalchemy'
39: app.config['SQLALCHEMY_DATABASE_URI'] = session_database_url
40: app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
41: app.config['SESSION_SQLALCHEMY'] = flask_sqlalchemy.SQLAlchemy(app)
42: flask_session.Session(app)
43:
44: #-----
45:
46: @app.before_request
47: def before_request():
48:     is_running_locally = '//localhost:' in flask.request.url_root
49:     is_using_https = flask.request.is_secure
50:     if (not is_running_locally) and (not is_using_https):
51:         url = flask.request.url.replace('http://', 'https://', 1)
52:         return flask.redirect(url, code=301)
53:     return None
54:
55: #-----
56:
57: @app.route('/', methods=['GET'])
58: @app.route('/index', methods=['GET'])
59: def index():
60:
61:     html_code = flask.render_template('index.html')
62:     response = flask.make_response(html_code)
63:     return response
64:
65: #-----

```

PennyAdmin09aHttps/penny.py (Page 2 of 3)

```

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

```

PennyAdmin09aHttps/penny.py (Page 3 of 3)

```

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

```

blank (Page 1 of 1)

1: This page is intentionally blank.

PennyAdmin09bHttps/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_session
12: import flask_sqlalchemy
13: import flask_wtf.csrf
14: import flask_talisman
15: import database
16: import auth
17:
18: from top import app
19:
20: #-----
21:
22: dotenv.load_dotenv()
23: app.secret_key = os.environ['APP_SECRET_KEY']
24: flask_wtf.csrf.CSRFProtect(app)
25: flask_talisman.Talisman(app)
26:
27: # Session configuration to store session data in the file system
28: # (simple, but not realistic for a deployed app):
29:
30: #app.config['SESSION_PERMANENT'] = False
31: #app.config['SESSION_TYPE'] = 'filesystem'
32: #flask_session.Session(app)
33:
34: # Session configuration to store session data in a database:
35:
36: session_database_url = os.getenv('SESSION_DATABASE_URL',
37:     'sqlite:///pennysessions.sqlite')
38: session_database_url = session_database_url.replace(
39:     'postgres://', 'postgresql://')
40: app.config['SESSION_PERMANENT'] = False
41: app.config['SESSION_TYPE'] = 'sqlalchemy'
42: app.config['SQLALCHEMY_DATABASE_URI'] = session_database_url
43: app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
44: app.config['SESSION_SQLALCHEMY'] = flask_sqlalchemy.SQLAlchemy(app)
45: flask_session.Session(app)
46:
47: #-----
48:
49: @app.route('/', methods=['GET'])
50: @app.route('/index', methods=['GET'])
51: def index():
52:
53:     html_code = flask.render_template('index.html')
54:     response = flask.make_response(html_code)
55:     return response
56:
57: #-----
58:
59: @app.route('/menu', methods=['GET'])
60: def menu():
61:
62:     auth.authenticate()
63:     username = auth.get_username()
64:
65:     is_authorized = database.is_authorized(username)

```

PennyAdmin09bHttps/penny.py (Page 2 of 3)

```

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

```

PennyAdmin09bHttps/penny.py (Page 3 of 3)

```

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

```

blank (Page 1 of 1)

1: This page is intentionally blank.

PennyAdmin09cHttpsLocal/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('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('Port must be an integer.', file=sys.stderr)
21:         sys.exit(1)
22:
23:     try:
24:         penny.app.run(host='0.0.0.0', port=port, debug=True,
25:                        ssl_context=('cert.pem', 'key.pem'))
26:     except Exception as ex:
27:         print(ex, file=sys.stderr)
28:         sys.exit(1)
29:
30: if __name__ == '__main__':
31:     main()

```

blank (Page 1 of 1)

1: This page is intentionally blank.