**DOCUMENTATION**

ITEC649-45511322

Author: L Dhanya Himaja and Steve Cassidy

The application consists of database.py, main.py, interface.py,users.py,positions,json,static and views.

* The database.py module contains code to create the database tables consisting of sample data.
* Main.py holds the actual script of the application.
* Interface.py consists of queries to retrieve and store positions in the database for frontend interface.
* Users.py contains queries for logging in, logging out and positions.
* Views consists templates of base, index, index\_login, error\_login html files.
* Static consists sample CSS and html files.

Initially we should run the file main.py to create the database. Also to reset database to a known state, we can re-run main.py.

Main.py :

.

* “/positions/<jobid>”

In this function, position data is sent to the positions.html page by calling position\_get function from interface.py . A template is rendered which displays the data of the respective job id. In URL :/positions/jobid , jobid is the id of position.

* '/login', method='POST'

check\_login: This function checks the Username and Password from the user database. A sessionid is then created for the user and index.html page is rendered, Otherwise a session is created which redirects to “/” .

* “/post”

Appen\_database: The user who is logged in can now add new positions in this function using position\_add() from interface.py. This gets redirected to “/”.

* “/logout”

Logout: Here session gets terminated after retrieving username from session.

The database contains the following tables:

* Users: nick (user nickname), password, avatar are the fields of users table.
* sessions: sessionid, usernick( usernick is a reference to the nick field in the users table) are the fields of session table.
* posts: id, timestamp, usernick, content are fields of table posts.

Positions table: The id field of positions table auto-increments integer on inserting a row. The timestamp field defaults to the current time and date as: ‘2018-11-03 05:26:53’.

* Home and about Page:

When I load the home page I see the title text "Welcome to Jobs" on the front end. The code for this is implemented in the index.html and login\_index files .Also, in home page I see a link “About” which takes me to about page on clicking it. The logic for this is implemented in index.html and login\_index.html. The “About” page contains “Jobs is a new, exciting, job posting service like nothing you've seen before!”.

* Positions In Home Page

On loading the home page (URL /) as a visitor to the site one can see a list of up to 10 positions in order of their timestamps, most recent first. In each position we can see the position timestamp, title, owner and the first 100 characters of the text of the position description. At the end of each position description there is a link to view the entire position named "Read More" and the url /positions/<id>where id is the position id. The recent 10 positions are fetched from database as a call to position\_list .Using for loop which gets executed 10 times as the limit given is 10 , the list of 10 positions values are implemented in index.html by parsing the value of ‘positions ’ from function in main.py.

* Position Page

On clicking read more in the home page, the page gets directed to URL /positions/id where we can see full job description .This is implemented in positions.html file. The 'description' value is parsed from main.py to display the complete job description.

The position page should contain all of the database fields for the position, laid out in a readable way. It should also contain a link to return to the main page.

* Login Form and Logging In

When the home page gets loaded, I see a form with text boxes for nick and password and a button to Login. The form will has id 'loginform' and the fields used are 'nick' and 'password'. The login is performed by the login action ‘/login.’On entering user nickname(Bobalooba) and password (bob) as a registered user and by clicking login button , the page gets redirected to , the login form gets replaced with a picture and a message pops up which reads”Logged in as Bobalooba” with a button adjacent labbled as “Logout”. On successfully logging in , the response redirects the user to home page. This response holds a cookie with sessionid name which holds the user usernick appended by the keyword “uses”. The logout button has an id “logoutform” in the form with an input submit named logout .

Login Failed

* On entering Incorrect password and click on the Login button, the response page will show the message "Login Failed, please try again". This page will contain another login form. Implementation for this is done in error\_login.html file
* I can post a new job as a registered user by filling out a form on the main page to create a new job listing (position), when I submit the form I am redirected to the main page and my new position appears in the list. The form for posting a new position will have the id postform and its action will be in the URL /post
* Logging Out

If I click on the Logout button , I get into the site home page in response and again the login form is displayed as the session gets deleted.

Back-end:

Interface.py

Here, all the connection are returned by the sqlite3.connect function.

* position\_list

position\_list(db, limit=10): This function returns a list of position tuples from database consisting : (id, timestamp, owner, title, location, company, description).Messages are returned in reverse order of the timestamp.

* position\_get

position\_get(db, id): This function returns a tuple containing (id, timestamp, owner, title, location, company, description) which represents the position with the given id, The function returns None if the id fails to match the record in the databse.

* position\_add

position\_add(db, usernick, title, location, company, description). This functionality adds the new position to the database. The function returns False If usernick fails to match user of users table and no new record will be created. True is returned if record gets added.

Users.py

This file consists of 4 functions which deal with users authentication and managing user sessions.

* check\_login

check\_login :This function returns false if the password in incorrect and true otherwise. The password here is stored as a one-way hash in the database. The function password\_hash(text) is used to create a one-way hash from password.

* generate\_session

generate\_session :this function returns none, if nick doesn't correspond to an existing user. If this user does not have an entry in the sessions table, a new session is created. The existing session id is retrieved, if the session already exists .A cookie in the Bottle response with the name sessionid is created by the function and the sessionid is returned.

* delete\_session

delete\_sessions :This function deletes all entries for user from sessions table.

* session\_user

session\_user : This function returns none if none and if the user is found, it returns the name of the user logged in . It so does this by finding session id from cookie and uses it to find user in the sessions table.

* insert\_session

insert\_session: This function inserts session for user in the session table if session is not present and returns the inserted sessionid.

* user\_present

user\_present: This function looks for this user in the user table. Returns false if user does not exist, else returns true.