Resume Critic - Documentation

David Mulatti, Kyle Petrozzi

1 Purpose

The purpose of this website is to have a place for University of Windsor students seeking employment to be able to upload their resume in PDF format and have it critiqued by their peers. Users can provide feedback in the form of star ratings or comments. The users can upload updated resumes as often as they like, and gain new feedback on it. The resumes are all public, but only registered users can rate and comment.

2 Installation

To install the website, simply extract the source code into your *public_html* folder. Then, replace the values in *dbaccess.php* with the appropriate information for the database you'd like to use, and /assets/recaptcha/secret.php with your own secret key provided by Google's reCAPTCHA service (one is provided with the site files). Lastly, navigate to "/admin/create_db.php" to create the database, then navigate to "/admin/populate_db.php" to populate the database with four test users, and an admin account.

Login Info

User Name	Password
admin	password
testone	password
testtwo	password
testthree	password
testfour	password

For a working demo, this site is live at 334.mulatti.com.

3 Database Design

The database for this site contains three tables: users, comments, and ratings. The users table stores user info, including uwinid, hashed password, full name, rating on their resume, resume upload date, description for their resume, and a boolean indicating whether they have uploaded a resume or not.

The comments table stores an auto incrementing comment id, the uwinid of the commenter, the uwinid of the user the comment is for, the date of the comment, and the comment itself.

The ratings table is almost identical to the comments table, but does not store the date, and is used to store ratings represented as a double rather than comments.

4 Site Walkthrough

4.1 Design and CSS

Bootstrap is used for the design of this site, with the assistance of a kickoff template provided by maxdesign [1]. There are modifications made to this, located in /assets/css/styles.css.

4.2 Admin Control Panel

When logged in as admin, an "Admin" option appears in the site header that will allow one to access the admin control panel. In this, the admin can view all of the tables in the database, edit any user, delete any comment, drop the database, create the database and populate the database with the test data shown above.

4.3 Header

The header is included in every page on the website. It contains a *session_start()*, the head containing the stylesheet, and jQuery and Bootstrap scripts. It also contains the site header itself, which is dynamic and changes if a user is logged in or not, and if an admin is logged in. Also, if *$headextra* is set, it will echo that data in the head, allowing additional scripts to be loaded on pages that need them.

If a user is not logged in, the "Upload Resume" button is replaced with a tooltip that appears when it is hovered over, notifying the user that they must be logged in to upload a resume. There is also a "Create Account" and "Contact Us" button, along with text fields for a user to log in.

If a user is logged in, the "Upload Resume" button becomes active and will redirect the user to the upload resume page. The "Create Account" button changes to a "Edit User Info" button, and the login fields are replaced by a welcome message and logout button. If the user logged in is an admin, there is an additional "Admin" button added to the site header that will redirect to the Admin Control Panel.

4.4 Footer

The footer is included in every page, and is simply the closing ⟨body⟩ and ⟨html⟩ tags, as well as a centered message including an emoji. The emoji is generated using Twitter's *twemoji* plugin [2].

4.5 Create Account

Here, there is a is a form that takes a uwinid, password, and full name to create a new account. There is also a captcha, provided using Google's reCAPTCHA api, to verify that the person signing up for an account is indeed human. The form uses the jQuery Validation plugin [3] to validate all the fields, including AJAX calls to the database to ensure that a username is not taken. It will also ensure that the password is at least 8 characters long, and that the full name only has alphanumeric characters in it.

On submission, all fields are sent as POST variables to *createaccount_go.php* which will insert a new user into the database. The password is hashed using the *password_hash()* function, provided by ircmaxwell [4].

4.6 Login and Logout

The login.php is relatively simple. It compares the password sent within the form to the hashed password stored in the database associated with the appropriate uwinid, using the *password_verify()* function. If it matches, it sets the *logged_in* session variable to 1, and then redirects back to the

index page. It also checks if the user logging in is the admin, and if so, changes the session variable to 3. It also has an error page if the wrong username or password is entered.

4.7 Edit User

The edit user link appears in the header whenever a user is logged in. A user can then edit their Full Name or their password. If an admin is logged in, then the edit user page is extended, allowing the admin to change many more options for any user they wish.

4.8 Resume Upload

This page is used to upload/edit your own resume page. If you are logged in you can add or update a resume and description. If you already have a resume uploaded, it shows it to you as an embedded object, and auto fills the description with your last description used. When you press the upload button it redirects to upload.php. This will check if the file uploaded is a application/pdf MIME type, if it comes from the POST function, and finally if it is not too large of a file. If all of these checks pass, then it saves the file as *uwinid*.pdf, *uwinid* being the username of the current user, in a "resume" folder located in the site root directory. It also updates the upload date, the *has_uploaded* boolean, and the description columns within the database to corresponding to the user who uploaded. It also will add a comment to the resume, notifying that a new resume has been uploaded. It then redirects back to the resume viewer page on success, and to an error page if something has failed.

4.9 Resume Viewer

When a name on the homepage is clicked, the user is brought to a page which displays the resume as an embedded object on the left, and the description, upload date, and comments on the resume on the right. Above the resume there is a current star rating of the resume, gathered by what the users have voted.

If the user is logged in, below the comments there is a textarea to add their own comment, and an option to rate the resume out of five stars. The star rating system is powered by a jQuery plugin for bootstrap made by GitHub user *kartik-v* [5].

4.10 Contact Us

The "Contact Us" page displays our names, some information about ourselves, and our email addresses. There is also a link to the GitHub page that our code is available on, and a form including a TinyMCE [6] editor to send an email directly to us using the PHP *mail()* function. In addition, if a user is logged in, their name and email address are automatically filled in.

On submission of the contact form, the user is redirected to *contact_go.php* which takes the POST variables from the form, validates the captcha, and processes the message given from the TinyMCE editor through XSLT to strip out everything that isn't a ⟨div⟩, ⟨strong⟩, or ⟨em⟩ tag. In addition, any ⟨a⟩ tag is stripped if the URL is not prefixed with "334.mulatti.com".

5 Security

Every database query that requires user input is made using prepared statements and the mysqli extension. All form data is verified using jQuery validate, and again by checking the input in the PHP code that processes it.

The admin pages can only be accessed when an admin is logged in, and will *die()* otherwise. The *create_db.php* and *populate_db.php* pages do not have this restriction, so that it is easy to set up the database when installing the site. However, if these try to be ran a second time, nothing will happen since the queries to create the tables have the "IF NOT EXISTS" constraint, and *populate_db.php* will throw errors because it is trying to create duplicate entries. Thus, restricting these access of these two pages only complicates the installation of the website.

References

[1] russmaxdesign, *Bootstrap Kickoff Template*
Available at: https://github.com/russmaxdesign/01-bootstrap-kickoff-template

[2] Twitter, *Twitter Emoji For Everyone*
Available at: https://github.com/twitter/twemoji

[3] jQuery Validation
Available at: http://jqueryvalidation.org/

[4] ircmaxwell, *password_compat*
Available at: https://github.com/ircmaxell/password_compat

[5] kartik-v, *bootstrap-star-rating*
Available at: https://github.com/kartik-v/bootstrap-star-rating

[6] TinyMCE
Available at: https://www.tinymce.com/