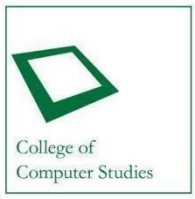

De La Salle University

College of Computer Studies
Center for Complexity and Emerging Technologies

Project Quiver

Technical Manual

Team Members Acorda, Victoria Angela

Amadora, Angelo John

Fernandez, Ryan Austin

Date Submitted August 16, 2016

Table of Contents

I. Executive Summary

Project Quiver is a site for showcasing projects from the College of Computer Studies. It was developed using PHP which runs on an Apache Server running PHP 5.6.8. This document details the folder structure, MVC architecture details, installation details, security policies, class and file specifications.

Project Quiver was developed under the Center for Complexity and Emerging Technologies (COMET) under its Civic Services thrust which aims to apply the research performed by the other thrusts or to simply develop systems that can be readily used in the real world.

Project Quiver is the intellectual property of Laurenz Tolentino, a graduate of Bachelor of Science, Major in Software Technology from De La Salle University – Manila. It was developed by Austin Fernandez, Gela Acorda, and Angelo Amadora, all undergraduates of the same degree program in the same university as well as COMET members.

II. Technical Specifications and Architecture

Project Quiver followed the MVC architecture but without using any external frameworks. The site was developed in PHP and runs on the Apache Server running PHP 5.6.8. This section will describe the folder structure; the model, view, and controller; installation details; and security guidelines.

Folder Structure

Project Quiver's root folder contains a few subfolders. "assets" contains any images used by the site. "css" contains all stylesheets. "documentations" contains any documentation including the Technical Manual, DB Models, and User Manual. "font" contains external fonts used by the site. "includes" includes the model and controller files. "js" contains any javascript files. "materialize" contains any css or js files used by the Materialize UI framework. "res" contains the admin-side files. "uploads" contains any files uploaded to the server.

Inside the "res" folder, for the admin side, this folder also contains "assets","css","font","includes","js", and "materialize" folders. It also contains a "commons" folder which includes header and footer php files.

MVC Architecture

Model

The "includes" folder contains all the model files. The globals.php file defines constants used by the system. The class.database.php file handles the database access, which is included by the main-functions.php file, which also includes the globals.php file. All future model files must then include the main-functions.php file or some file that includes the main-functions.php file.

View

All view files are in either the root directory or the res folder. At the beginning of each file, before the first non-PHP script line, all pertinent data must be retrieved. Since no framework is used, the html is constructed using PHP conditional statements and iterative statements.

Controller

All forms must have a request parameter (<input type="hidden" name="request"/>) that contains the request to which the form is to be sent. All form-actions are directed to includes/controller.php.
controller.php then has a switch statement that switches based on the value of the request parameter. Additional request types can be added by adding a new switch-case.

Installation

To install the system onto a working Apache Server:

1. Place the ProjectQuiver folder in the server's root directory where projects are placed. The root Project Quiver folder will henceforth be referred to as $ROOT.
2. Make sure a working database server is ready.
3. Run the database schema file at $ROOT/documentations/COMET_ProjectQuiver_EmptyDBDump.sql
4. Go to $ROOT/includes/globals.php and $ROOT/res/includes/globals.php and input the correct information pertaining to the database server.
5. Make sure the server is running HTTPS
6. Try going to the domain assigned to the site. The front page should show up.

7. To access the admin side, go to the domain/res

Security Policies

- **Authentication**
  - Passwords are hashed using the blowfish BCrypt algorithm.
  - When a session is idle for 24 hours, the user is logged out.
  - Any session can only last one week or 168 hours.
  - A user can attempt to log in 5 times incorrectly before they are logged out for fifteen minutes.
  - The query for the username password only uses the username as a key.
  - All default accounts were removed from the database.
  - Passwords have an enforced complexity standard that each password has at least one uppercase letter, one lowercase letter, 8 characters, one special character, and one number.
  - Accounts expire after one year of inactivity.
  - Reauthentication is enforced when an admin creates a new admin account.
- **Authorization**
  - The authorization matrix is in the pq_role table in the database.
  - Whenever a user loads a page or enters the controller, their authorization is rechecked.
- **Session Management**
  - The site is forced to HTTPS. No sessions or cookies persist unless the server is running HTTPS.
  - URL rewriting must be disabled on the server configuration.
  - Sessions are refreshed on session restore from cookie, login, and logout.
  - Cookies are set to http only and secure in the server and the code.
  - The custom session token uses the IP address in the hashing (which is a SHA256 hash).
  - The cookie is set to expire after one week.
- **Data Validation and XSS**
  - Validation is always performed using regex in the view (javascript) and the controller (controller.php).
  - All user-provided output is encoded using htmlspecialchars().
  - All queries are parametrized.
  - CSP, X-XSS-Protection, X-Content-Type-Options, Content-Type headers are defined in main-functions.php.
- **Auditing and Logging**
  - All important events (create,update,and delete) are audited, including the user id, ip address, timestamp, and the action.
- **Cross Site Request Forgery and Clickjacking**
  - Hidden form tokens are included in all GET and POST requests.
  - X-Frame-Options header is included in main-functions.php.

III. Class and File Documentation

PHP

Class	Database @author Austin Fernandez @20151031 This class manages database access.
Properties	$db – local db connection
Methods	__construct constructs a basic db connection @param $dbType - type of database @param $dbHost - host address of db server @param $dbName - schema name @param $dbUser - username on server @param $dbPass - password for corresponding user on server
	query executes an sql statement over the database @param $type type of statement (SELECT,INSERT,UPDATE) @param $sql sql statement @param $param [optional] associative array of parameters @return object comprising of status -> true if successful, false otherwise data -> array of associative arrays with all the data count -> number of rows returned error -> error returned

File	audit-functions.php @author Austin Fernandez @20160810 This file is responsible for auditing user actions.
Functions	audit_add adds an audit to the db @param @action action of the user in the audit @return 1 if successful, 0 otherwise

File	globals.php @author Austin Fernandez @20160813 This file defines globals for the system.
Constants	DBTYPE – type of database used by the system
	DBHOST – host address of the db server
	DBNAME – schema name (“db_project_quiver”)
	DBUSER – username in server
	DBPASS – password for username
	MAX_LOGIN_ATTEMPTS – maximum number of login attempts before lockout (5)

File	image-functions.php @author Austin Fernandez @20160609 This file manages image uploads
Functions	img_upload This uploads a list of photos to the given foldername under the uploads folder. @param $folderName name of folder to save images in @param $images list of images, each with name - name of image size - size of image tmp_name - temporary location of image @return array of paths to the images or FALSE if an error occured

File	main-functions.php @author Austin Fernandez @20151102 This file handles all unclasssified functions. HTTP Headers are also added here
HTTP Headers added	X-Frame-Options - sameorigin
	Content-Security-Policy - default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval' https://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jquery.min.js; object-src 'self'; style-src 'self' 'unsafe-inline' https://fonts.googleapis.com/; img-src 'self'; form-action 'self'; media-src 'self'; font-src 'self' https://fonts.gstatic.com/ https://fonts.googleapis.com/ https://applesocial.s3.amazonaws.com/ ; plugin-types application/pdf application/x-shockwave-flash; reflected-xss block;
	X-Content-Security-Policy - default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval' https://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jquery.min.js; object-src 'self'; style-src 'self' 'unsafe-inline' https://fonts.googleapis.com/; img-src 'self'; form-action 'self'; media-src 'self'; font-src 'self' https://fonts.gstatic.com/ https://fonts.googleapis.com/ https://applesocial.s3.amazonaws.com/ ; plugin-types application/pdf application/x-shockwave-flash; reflected-xss block;");
	X-Webkit-CSP - default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval' https://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jquery.min.js; object-src 'self'; style-src 'self' 'unsafe-inline' https://fonts.googleapis.com/; img-src 'self'; form-action 'self'; media-src 'self'; font-src 'self' https://fonts.gstatic.com/ https://fonts.googleapis.com/ https://applesocial.s3.amazonaws.com/ ; plugin-types application/pdf application/x-shockwave-flash; reflected-xss block;
	X-XSS-Protection - 1;mode=block
	X-Content-Type-Options – nosniff
	Content-Type - text/html; charset=utf-8
Functions	format_date converts YYYY-MM-DD format to Month Day, Year @param $date date to convert @return date in correct format
	validateDate validates that a date is in YYYY-MM-DD format

	@param $date date to validate @return true if valid, false otherwise
	validateEmail validates an email address @param $email email to validate @return true if valid, false otherwise
	validateDLSUEmail validates an email address @param $email email to validate @return true if valid, false otherwise

File	project-functions.php @author Austin Fernandez @20150722 This file handles all project-related functions.
Functions	proj_get_all returns all projects that have been judged @return all projects in a list with each element having name - name of project class - classification of project abstract - abstract of project description - description of project review - review of project reviewer - name of reviewer
	proj_get_pending returns all pending projects @return list of pending projects with name - name of project class - classification of project abstract - abstract of project description - description of project review - review of project reviewer - name of reviewer
	proj_get_best returns the top projects for displaying in the Quiver site @return list of top projects with name - name of project class - classification of project abstract - abstract of project description - description of project review - review of project reviewer - name of reviewer
	proj_get gets a single project @param $id id of project @return project associative array with name - name of project

	class - classification of project abstract - abstract of project description - description of project review - review of project reviewer - name of reviewer
	proj_review adds a review to a project @param $id id of project @param $reviewer id of reviewer @param $review review text @param $grades list of grades @param $recogs list of recognitions @return TRUE if successful and FALSE otherwise
	proj_add adds a project to the database @param $name name of project @param $class classification of project @param $abstract abstract of project @param $desc description of project @param $students list of students with idNo id number fName first name lName last name email email address @param $tags list of tags @return id of project if successful and FALSE otherwise
	proj_add_images adds images to a project in the database @param $id id of project @param $images list of relative urls for images @return true if successful, false otherwise

File	security-functions.php * @author Austin Fernandez * @20160810 * This file handles security related operations.
Functions	checkAuth checks if a certain feature is allowed by the current user @param $feature feature to be tested @return 1 if allowed and 0 otherwise
	genHash generates a hash using ip address as client-side control @param $ip ip address of client @return hashed token
	genToken forcefully generates a new token @param $ip ip address of client

	restoreToken restores the session token @return new session token
	checkToken checks if given token is equal to client side token @param $token token to chec @return true if tokens match, false otherwise

File	student-functions.php @author Austin Fernandez @20160705 This file handles student related operations.
Functions	student_add adds a student to the database @param $idNo id number @param $fName first name @param $lName last name @param $email email address @param id of student if successful, FALSE otherwise
	student_get gets a students' details @param $idNo id number @return associative array of the student with id - id of student in database idNo - id number fName - first name lName - last name email - email address

File	user-functions.php @author Austin Fernandez @20160810 This function manages user data in the database.
Functions	usr_add adds a user to the database @param $email email address of user @param $password password of user @param $fname first name of user @param $lname last name of user @param $usrType type of user @return true if success and false otherwise
	usr_check checks if a password is correct for an account @param $email email address of account @param $password password to check @return id of user if correct password, FALSE otherwise
	usr_get_session

	Returns the session user or null if none. If no user is set, it tries to use a cookie to restore a session @return session user object or NULL if none
	usr_get gets the details of one user @param $id id of user in database @return user containing id - id of user in db email - email address of user fName - first name of user lName - last name of user addProject - 1 if user can add project, 0 otherwise judgeProject - 1 if user can judge project, 0 otherwise createUser - 1 if user can create User, 0 otherwise deleteUser - 1 if user can delete User, 0 otherwise
	usr_get_by_email gets user details using email @param $email email address of user @return user containing id - id of user in db email - email address of user fName - first name of user lName - last name of user addProject - 1 if user can add project, 0 otherwise judgeProject - 1 if user can judge project, 0 otherwise createUser - 1 if user can create User, 0 otherwise deleteUser - 1 if user can delete User, 0 otherwise

Javascript

File	addProject.js @author Austin Fernandez @20160813 This file handles the add project screen. Main namespace is “addProject”
Functions	checkSubmit checks the form if all fields' values are valid. @return true if everything is valid, false otherwise

File	createAccount.js @author Austin Fernandez @20160813 This file handles the create account screen.
Functions	checkSubmit checks the form if all fields' values are valid. @return true if everything is valid, false otherwise

File	createAccount.js @author Austin Fernandez @20160813 This file handles the create account screen.
Functions	showError shows the given error message @param message message to be shown
	showMessage shows the given message @param message message to be shown
	appendMessage constructs a new message given the current message and an addition @param message current message @param add string to add @return new message
	checkPass checks if a password meets the standards i.e. has at least one uppercase letter, one lowercase letter, 8 characters, one special character, and one number @param pass password to check @return true if valid, error message otherwise