# Exercise Site –Project hand in

### Final date: 14th Nov to Friday 18th Nov 10am

Your project will be marked in class in the last week of the term

### Value: 40%

Please install all files on kate (make sure your database tables are populated with the required data). Also copy your project into the dropbox on M:\Assessments Swallow Box\IN612Web2\Assignment 4.

Your site must provide (at a minimum) the following functionality:

1. User registration
2. User login (multiple user site)
3. Main welcome screen
4. Data entry screen
5. Tabular view of all data entered for the logged in user
6. Life time summary statistics table for the logged in user
7. Calendar view coloured by activity
8. Graphs: current user’s last 7 days, current user’s last month or current month
9. Pie graph for the current user
10. Targets
11. Other motivating graphs or data- leader boards – automated comments
12. All users- last 7 days comparison
13. Other motivating graphs or statistics showing the logged in user vs. the other people

## Other Requirements

For testing and setup purposes you are required to have the following two files.

createTables this is where you create all the tables.

showTables this is for testing purposes you must show all fields and all records in all tables.

Hard code a user: login name **dale** password **test**

This gives me access to your site to check your progress.

Version control:

You must set up a version control system. This year we will be using GitBucket.

For the version control mark you must make a minimum of 2 submits per week starting on 31st October.

# Site -Coding Preparation Document

### Due date: 8/9th Nov

### Value: 10% of project mark

Please fill in all parts of this document ideally before you begin coding[[1]](#footnote-1).

1. List any functionality you intend to provide additional to that listed above

Track BMI – Will implement if I have time

1. List or sketch (just little boxes will do – just give each one a name so you can refer to it below) the individual screens you will provide in your site. Make sure that you have the necessary screens for all of the functionality required.
2. In the table below, list all the screens you are building, as named above. For each screen, give the name of the file which will produce the html for that screen. Name the controller file that includes these screens.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen | File name | Controller file name |  |
| Landing Page | LandingPage.html.php | LoginController.php |  |
| Login Page | LoginPage.html.php | LoginController.php |  |
| Register Page | RegisterPage.html.php | LoginController.php |  |
| Home Page | homepage.html.php | siteController.php |  |
| Enter Workout Page | enterWorkout.html.php | siteController.php |  |
| Add Activity | addActivity.html.php | siteController.php |  |
| Raw Data Page | rawData.html.php | siteController.php |  |
| User Graphs Page | userGraphs.html.php | siteController.php |  |
| User Tabular Data Page | userData.html.php | siteController.php |  |
| Friend Comparison Page | friendComparison.html.php | siteController.php |  |
| Track BMI page | Bmi.html.php | siteController.php |  |

1. List any additional utility files (i.e. files that don’t actually produce any screens, but just perform work). Examples are your dbConnect include file and any function libraries. For each listed file, briefly describe its purpose.

|  |  |
| --- | --- |
| File Name | File Purpose |
| connect.inc.php | Holds connection information |
| connectToServer.php | Connects to the server |
| error.html.php | Displays errors that occur |
| functions.php | Hold all functions used for the site |
| LoginController/  siteController | Controls flow of website |
| StyleSheet.css | Holds all css |
| Workouts.php | Creates and holds information for the creation of the Calendar |

1. In the table below, list each functional element you are going to provide that requires user input (e.g. elements 1, 2, 4, and 5 on the first page of the assignment handout). For each function listed, name the screen(s) the user will see when providing that input. For each screen, list *all* the input controls on the screen (don’t forget the submit buttons) with their type, name and value properties as they will appear in the generated <input...> tag.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function | Screen | Input Control Type | Input Control Name | Input Control Value |
| Logging In | LoginPage | text | emailLogin | Nil |
| Logging In | LoginPage | text | passwordLogin | Nil |
| Logging In | LoginPage | submit | LoginButton | Nil |
| Register | RegisterPage | text | fName | Nil |
| Register | RegisterPage | text | lName | Nil |
| Register | RegisterPage | text | email | Nil |
| Register | RegisterPage | number | height | Nil |
| Register | RegisterPage | password | password1 | Nil |
| Register | RegisterPage | password | password2 | Nil |
| Register | RegisterPage | text | secretCode | Nil |
| Register | RegisterPage | submit | RegisterButton | Submit |
| Add Activity | addActivityPage | Text | activityName | Nil |
| Add Activity | addActivityPage | Html5colorpicker | colourName | Nil |
| Add Activity | addActivityPage | Submit | submitActivityButton | Submit |
| Enter Workout | enterWorkout | Select | activityResult | Nil |
| Enter Workout | enterWorkout | Number | workoutDuration | Nil |
| Enter Workout | enterWorkout | Date | Datepicker | Nil |
| Enter Workout | enterWorkout | Textarea | workoutComment | Nil |
| Enter Workout | enterWorkout | Submit | submitWorkoutBtn | Submit Workout |
|  |  |  |  |  |

1. In the table below, list each functional element you are going to provide that requires user input. For each function listed, write out the mySQL query or queries that you will perform to support the functionality. Refer to the database tables described in your design document. Check that the controls you listed in Table 5 will enable you to perform all the queries listed here (pay special attention to how you will be able to tell which items are being added to the cart).

|  |  |
| --- | --- |
| Function | Queries |
| doInsert  \*Adds a user | $insertQuery = "INSERT INTO tblUser (firstName, lastName, email, height, userPassword) VALUES (:fName, :lName, :email, :height, :hash)";  $preparedStatement = $pdo->prepare($insertQuery);  $preparedStatement->bindParam(":fName", $cFName);  $preparedStatement->bindParam(":lName", $cLName);  $preparedStatement->bindParam(":email", $cEmail);  $preparedStatement->bindParam(":height", $cHeight);  $preparedStatement->bindParam(":hash", $hash);  $cFName = clean\_data($fName);  $cLName = clean\_data($lName);  $cEmail = clean\_data($email);  $cHeight = clean\_data($height);  $preparedStatement->execute(); |
| addActivity  \*Adds activity to the database | $insertQuery = "INSERT INTO tblActivity (activityName, activityColour) VALUES (:accName, :colour)";  $preparedStatement = $pdo->prepare($insertQuery);  $preparedStatement->bindParam(":accName", $cAccName);  $preparedStatement->bindParam(":colour", $colour);  $cAccName = clean\_data($accName);  $preparedStatement->execute(); |
| addWorkout  \*Adds workout to the database | $insertQuery = "INSERT INTO tblWorkout (userID, activityID, workoutDate, workoutMinutes, workoutComments) VALUES (:userID, :activityID, :workoutDate, :workoutDuration, :workoutComment )";  $preparedStatement = $pdo->prepare($insertQuery);  $preparedStatement->bindParam(":userID", $cUserID);  $preparedStatement->bindParam(":activityID", $activityID);  $preparedStatement->bindParam(":workoutDate", $workoutDate);  $preparedStatement->bindParam(":workoutDuration", $cWorkoutDuration);  $preparedStatement->bindParam(":workoutComment", $cWorkoutComment);  $cUserID = clean\_data($userID);  $cWorkoutDuration = clean\_data($workoutDuration);  $cWorkoutComment = (string)$workoutComment;  $cWorkoutComment = clean\_data($cWorkoutComment);  $preparedStatement->execute(); |

1. List any additional queries you will perform on your database that are not directly tied to the processing of user input. For each query, briefly describe its purpose.

|  |  |
| --- | --- |
| Queries | Purpose |
| findActivities  \*Provides a list of activities | $selectString = "SELECT \* FROM tblActivity";  $activityResult = $pdo->query($selectString); |
| findUsers  \*Finds all users in the database | $selectString = "SELECT \* FROM tblUser";  $userResult = $pdo->query($selectString); |
| findWorkouts  \*Finds all workouts in the database | $selectString = "SELECT \* FROM tblWorkout";  $workoutResult = $pdo->query($selectString); |
| findWeekWorkouts  \*finds all workouts for a single user in the last week | $selectString = "SELECT workoutDate, sum(workoutMinutes) as minutes FROM tblWorkout WHERE userID='$userID' AND workoutDate > DATE\_SUB(NOW(), INTERVAL 1 WEEK) GROUP BY(workoutDate)";  $workoutResult = $pdo->query($selectString); |
| findBMI  \*Finds all BMI entries | $selectString = "SELECT \* FROM tblBMI";  $BMIResult = $pdo->query($selectString); |

1. For each file you described in parts 3 and 4 above, list the data variables that will be stored in and/or retrieved from the $\_SESSION array.

|  |  |  |  |
| --- | --- | --- | --- |
| File name | Variables | Stored in $\_SESSION? | Retrieved from $\_SESSION? |
| Every File siteController refers to includes $\_SESSION | $\_SESSION[‘userID’]  $\_SESSION[‘firstName’] | userID  firstName | userID  firstName |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. For each file you described in parts 3 and 4 above, list all the data validation that you will perform. For each data element, give its local variable name, the name of the file which produces the html for the associated input control, the name of the data element in $\_POST (i.e. its associative index in the $\_POST array) and the type of validation to be performed. Give the php statement you will use to validate it or a description

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| File | Variable name | Data from what file? | Data from what element of $\_POST | Validation to be performed |
| Login Controller | $fName | RegisterPage | $\_POST["fName"] | $firstNameCriteria = "(^[a-zA-Z]{2,}$)"; |
| Login Controller | $lName | RegisterPage | $\_POST["lName"] | $lastNameCriteria = "(^[a-zA-Z]+$)"; |
| Login Controller | $email | RegisterPage | $\_POST["email"] | if (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) |
| Login Controller | $height | RegisterPage | $\_POST["height"] | $heightCriteria = "(^[0-9]{2,3}$)"; |
| Login Controller | $password | RegisterPage | $\_POST["password1"] | $passwordCriteria = "((?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[@#$%]).{6,20})"; |
| Site Controller | $actName | addActivity Page | $\_POST["activityName"] | $activityNameCriteria = "(^[a-zA-Z]{2,15}$)"; |
| Site Controller | $workoutComment | enterWorkout Page | $\_POST['workoutComment'] | Data is cleaned |
| Site Controller | $activityID  $workoutDate  $workoutDuration  $colourCode | enterWorkout  &  addActivity  Pages |  | User cannot enter text for these fields so no validation is needed |

\*All data that is text input gets cleaned.  
\*All data is sent to the database using prepared statements

1. Sketch your database model

1. During coding, things may come up which you didn’t think of when preparing this document. In that case, you may modify the document, but make it clear what was done in advance and what was done post hoc. Such changes must be kept to a minimum; this will be the natural result of working carefully through the document initially. [↑](#footnote-ref-1)