Fishing club Report

Introduction

Hello this is my report of my fishing club database project that was part of the dynamic web development Module as part of my 3rd year circulum at ATU Galway. The follow code was written in respects to a layout we were given last semester. I updated my code from last semester to allow for data encryption, a login as well as better use of sessions and header. I hope you enjoy my report and thank you for taking the time to read it.

SQL Section

The sql code For the fishing databases creates a database named fishing, with multiple tables to hold information related to the fishing club, with tables populated with sample data. The first few statements are designed to allow for fresh inputs when pasting the database into the sql console. The first few lines drop the fishing database if it already exists and create a new one with the utf8mb4 character set and collation. Another statement switches to the fishing database in an sql console window. The next Six statements drop any existing tables named fish, member, lakes, swims, catchreport, and user.

These tables will hold information about user authentication, fish, fishing club members, lakes, fishing swims, and catch reports. The next two statements set the starting value and increment for the auto\_increment column in each table to 10. The next statement creates the user table with columns for user id, email, and password. The id column is an auto-incrementing integer and the primary key of the table. The next statement creates the fish table with columns for fish species id, description, and details. The species id column is an auto-incrementing integer and the primary key of the table.

The next statement creates the member table with columns for member number, title, first name, last name, house number, street, town, county, postal code, phone number, mobile number, and email address. The member number column is an auto-incrementing integer and the primary key of the table. The next statement creates the lakes table with columns for lake id, name, number of swims, and features. The lake id column is an auto-incrementing integer and the primary key of the table. The next statement creates the swims table with columns for swim number, notes, and the lake id of the swim.

The swim number column is an auto-incrementing integer and the primary key of the table. The lake id column is a foreign key referencing the lakes table. The final statement creates the catch report table with columns for catch tag id, member number, swim number, fish species id, catch weight, and catch conditions. The catch tag id column is an auto-incrementing integer and the primary key of the table. The member number, swim number, and fish species id columns are foreign keys referencing the member, swims, and fish tables respectively.

The next ten statements insert sample data into the different tables throughout the database including the Fish, member, Swim, lake and catch report tables I did not include the user table as I was unsure how I would input encrypted data into it.

PHP and HTML Section

For the login pages I have a html page for login, logout and register with a small aspect of the home page also being dedicated to this for Php I used a file called query.php which starts by defining database connection parameters all of my php files do this so I will not be saying this for the remainder of the report.   
It then checks the value of the 'action' parameter passed through POST, and performs different actions based on its value. If the 'action' value is 'registration' for the registration page , the script retrieves the email and password from POST, encrypts the password using the password\_hash() function, and inserts the email and hashed password into the 'user' table in the database using an SQL query.   
If the 'action' value is 'login', the script retrieves the email and password from POST, retrieves the user with that email from the database using an SQL query, and compares the hashed password stored in the database with the entered password using the password\_verify() function. If the password is correct , a session is started and the user's email is stored in the session variable 'email', and the user is redirected to the 'home.php' page which will display the email of the new user in a welcome message. If the password is not correct , the user is redirected back to the 'login.php' page.

For the Insert page it has a Html page which displays the input form the user has to use to input the data into the database. The php script retrieves the form data sent via POST and assigns it to local variables. The 'mysqli\_query()' function is used to execute an SQL insert statement that inserts the data collected from the form into the applicable table. The 'mysqli\_affected\_rows()' function is used to check if the insert statement has affected any rows in the database. If any rows are affected, a success message is displayed. Otherwise, an error message is displayed and the 'mysqli\_error()' function will display. Finally, the 'mysqli\_close()' function is used to close the connection to the MySQL database.

For the select page I used a php script retrieves data from any of the tables depending on the Select page. The retrieved data is then displayed in a table with an amount of columns to describe the data from the tables. This part is very basic and is also used when looking at the delete table so I will go more into detail in that description

For the Delete pages I used a php script retrieves data from any of the tables depending on the delete page. The retrieved data is then displayed in a table with an amount of columns to describe the data from the tables + 1 for a delete column. The while loop iterates through each row of the table and displays the data in the corresponding columns. In the Delete column, there is a link to the same page with a query string containing the ID of the row to be deleted. When the link is clicked, the script checks if the delete parameter is set in the query string, and if so, deletes the row with the corresponding ID from the table in the database. Finally, the header() function is used to redirect the user back to the same page after the deletion is complete. The confirmation dialog box pops up when the user clicks on the Delete hyperlink to confirm if the user wants to delete the row or not.

For the Update page I used a html form to allow the user to enter the primary key of the data they want to change aswell as a specific part of the table they could update depending on the primary key they entered. The PHP script first checks if the user has submitted data into the previous HTML form by checking if the "submit" button has been pressed. If the button has been pressed, the script extracts the values submitted in the fields from the form.

Next, the script checks if a record with the specified Primary key exists in the table. If it exists, the script updates the table field of the record with the specified Primary key. If the update is successful, the script displays a success message. If the record does not exist, the script displays an error message. If the user has not submitted any data or if any of the required fields are empty, the script displays a message asking the user to enter all values. Finally, the script displays an HTML form for entering the Primary key and data fields for updating the record Further.

Conclusion   
I found this project to be very interesting throughout my time doing it I think I would like to proceed my career in future to a similar field or idea since making databases is quite interesting and quite important in modern society